POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Machine Learning for the Prediction of Insect Infestations in Hop Fields

Supervisors
Prof. Giovanni Squillero
Prof. Sandro Cumani
Dr. Alberto Tonda

Candidate
Zi Wang
matricola: 253012

Academic year 2019-2020

Abstract

European Corn Borer (ECB) is an insect that has been known as a hop pest, its infestations are one of the major problems for hop cultivation. In this thesis, several supervised machine learning techniques are applied on the data collected by the Slovenian Institute of Hop Research and Brewing for 18 years to predict the number of ECB, and the effectiveness of these techniques is compared and analyzed. The collected data consists of two parts: The first part is weather condition data which contains temperature, relative humidity, and precipitation from 8 pm to 6 am each day from April 30 to September 28, each year from 1999 to 2017; the second part is the number of ECBs captured at night on the corresponding date. The first part data is considered as the input object, while the second is considered as the desired output value. A thorough examination allowed us to conclude that regression methods are not the solution: a svm model with the help of manually added historical weather conditions information, the best out of twenty models, got an R^2 score of 0.04, only slightly better than a purely random guess. On the contrary, the work shows that the regression problem could be turned into a classification problem: predicting if the number of ECB captured is greater than a threshold. Moreover, statistics information of history can be added to the data set. Ten different kinds of classifiers and three different kinds of ways to add history information have been evaluated. Unfortunately, despite a significant improvement, results are still not accurate enough. The work shows a promising path towards the solution of this industrially-relevant problem, pinpointing deficiencies in the currently-available input data.

Summary

Introduction of the problem

European Union (EU) is an important hub of the global hop market. In Europe, the larvae of several moths and butterflies attack hops but only the European corn borer (ECB) reaches damaging levels frequently. The European corn borer, Ostrinia nubilalis, for example, was nicknamed the "billion dollar bug" because it cost growers over a billion dollars annually in insecticides and lost crop yields. Given that it is a short time for ECB larvae to be exposed to natural enemies or to insecticides, the optimal time window of ECB control methods is also short. Thus, a reliable prediction of ECB's population would improve the efficiency and the effectivity of ECB control.

To make a reliable prediction, a suitable mathematical model must be created. As we know, machine learning algorithms are able to build a mathematical model to make predictions by given sample data known as "training data". Nowadays, many software suites containing various machine learning algorithms are developed, which makes it easier to apply those algorithms on computers.

The goal of the thesis

In this thesis, several machine learning techniques are applied to the data of weather conditions and ECB's number and their performances are evaluated. The goal of this thesis is to explore a way to construct a reliable model that can predict the number of ECB by weather conditions.

Dataset

The data set is collected by the Slovenian Institute of Hop Research and Brewing from 8 pm to 6 am each day from April 30 to September 28, each year from 1999 to 2017, which includes 2875 different observations. It consists of weather conditions, the number of ECB captured during the night and the

corresponding date, where the number of ECB captured is the target variable that should be predicted by the model.

The weather conditions data contains temperature, relative humidity and precipitation of each day from 8 pm to 6 am. The number of ECBs captured ranges from 0 to 178 where 1515 observations (52.7%) are 0 and 2186 observations (76.0%) are from 0 to 4. As the number of ECBs is the target variable, the data set is highly unbalanced. Together with 78 incomplete observations and 13 missing observations, it makes the prediction difficult.

Experiments on the regression methods

At the beginning, the regression methods are evaluated. To quantify the quality of predictions the following three metrics is used: R^2 score, mean squared error (MSE) and explained variance score (EV) . For R^2 score and EV, The best possible score is 1.0 and lower values means it's a worse model. For MSE, a good model comes with a small MSE value.

First, all algorithms are run with the default hyperparameters. Unsurprisingly, all algorithms have a poor performance. The best 3 models are *ElasticNetCV*, *BayesianRidge* and *LassoCV* (their data set is denoted as *raw feat.* in Table 1).

Given that the best hyperparameters of *ElasticNetCV* and *LassoCV* are automatically select, *BayesianRidge* is the only model that should be tuned. After tuning, there is no improvement comparing to the model with the default hyperparameters (Table 1: *BayesianRidge(tuned)*, *raw feat.*).

Then 4 different ways to add historical weather information are tested. Training with raw data AND 4 different kind of historical information is denoted as *f.s. 1*, *f.s 13*, *f.s. 2*, *f.s. 3* respectively and training without the raw data is denoted as *(the name of the way) + ONLY*.

The best two models are *ExtraTreesRegressor* and *RandomForestRegressor* with the data set *f.s 13 ONLY*

After tuning, their performances are improved a little bit, but still not good enough (Table 1: *ExtraTreesRegressor(tuned)*, *f.s 13 ONLY* and *RandomForestRegressor(tuned)*, *f.s 13 ONLY*).

Another problem is 10-fold cross-validation shuffles all the data and probably makes the prediction easier. So models with the default parameters are evaluated with "leave one year out", a way to split the training set and the test set, which is closer to the reality than 10-fold. The results show that the best model is *NuSVR* with *f.s 1 ONLY* (Table 1: *NuSVR*, *f.s 1 ONLY*). All models perform poorly.

Experiments on the classification methods

Given the significant decrease of the quality of prediction, the regression problem is reduced to classification problem to reduce the difficulty of the problem. To do that a threshold is set to 10. All the number of ECB whose value is larger than or equal to 10 is replaced with label 1 (positive), otherwise it is replaced with label 0 (negative).

Four metrics (balanced accuracy, precision, sensitivity and specificity, are denoted as Accu., Pre., Sen., Spe. respectively) are used. All of them are from zero to one and a larger value means the model is better.

Models with the default parameters and both of 10-fold and "leave one year out" are evaluated. When 10-fold is used, *ExtraTreesClassifier* with *f.s 13 ONLY* and *RandomForestClassifier* with *f.s 13 ONLY* have the best performances among all models (Table 1: *ExtraTreesClassifier*, *f.s 13 ONLY* and *RandomForestClassifier*, *f.s 13 ONLY*). When "leave one year out" is used, *GaussianNB* with *f.s. 1 ONLY*, *f.s 13 ONLY* has the the highest balanced accuracy and very similar performance, however, *AdaBoostClassifier* with *f.s. 13* has a slightly lower balanced accuracy, higher precision, specificity and lower sensitivity (Table 1: *GaussianNB,f.s. 1 ONLY*, *f.s 13 ONLY* and *AdaBoostClassifier*, *f.s. 13*). The results also show that 10-fold makes the prediction easier.

Conclusions

Comparing to the beginning, where all models just give predictions that are almost close to random guesses, there's a certain degree of improvement with the help of adding the weather condition history and reducing the problem to the classification problem. But in order to be used by the hop farmers, there are still lots of jobs to be done.

Based on the evaluations that have been done, collecting more data could be helpful. Trying other kinds of learning algorithms is also a choice. Those algorithms which can deal with entire sequences of data, such as LSTM, may have advantages. Combining two different kinds of models may also be helpful. Finally, adding the number of captured ECB in the last few days to the training process of models may be a good attempt.

Acknowledgements

I would like to express my sincere gratitude to French National Institute for Agriculture, Food, and Environment (INRAE) for the continuous support,

Model	Dataset	R^2	MSE	EV
ElasticNetCV	raw feat.	0.08	133.30	0.09
BayesianRidge	raw feat.	0.08	133.37	0.09
LassoCV	raw feat.	0.08	133.40	0.09
BayesianRidge(tuned)	raw feat.	0.08	133.37	0.09
ExtraTreesRegressor	f.s 13 ONLY	0.46	65.16	0.47
RandomForestRegressor	f.s 13 ONLY	0.39	73.41	0.40
ExtraTreesRegressor(tuned)	f.s 13 ONLY	0.46	64.98	0.47
RandomForestRegressor(tuned)	f.s 13 ONLY	0.42	71.02	0.43
NuSVR	f.s 1 ONLY	0.04	164.67	0.12

Model	Dataset	Accu.	Pre.	Sen.	Spe.
ExtraTreesClassifier	f.s. 13 ONLY	0.74	0.74	0.53	0.94
RandomForestClassifier	f.s. 13 ONLY	0.73	0.75	0.51	0.94
GaussianNB	f.s. 1 ONLY	0.68	0.37	0.85	0.51
GaussianNB	f.s. 13 ONLY	0.68	0.38	0.84	0.51
AdaBoostClassifier	f.s. 13	0.66	0.54	0.44	0.87

Table 1. Experiments results

and Slovenian Institute of Hop Research and Brewing for creating the data set.

Acknowledgements

I would like to express my sincere thanks to my family, my parents for giving birth to me and supporting me throughout my life.

Contents

1 Introduction 9
- 1.1 Hop . . . 9
- 1.2 European corn borer (Ostrinia nubilalis) . . . 9

2 Materials and Methods 11
- 2.1 Data set . . . 11
  - 2.1.1 Missing values . . . 11
  - 2.1.2 Some characters of the data set . . . 12
- 2.2 Regression methods . . . 15
  - 2.2.1 PLS regression . . . 15
  - 2.2.2 Ensemble method . . . 15
  - 2.2.3 Bayesian Regression . . . 17
  - 2.2.4 Linear Model . . . 18
  - 2.2.5 Nearest Neighbors Regression . . . 20
  - 2.2.6 Support Vector Machine . . . 20
  - 2.2.7 Decision Tree Regression . . . 21
  - 2.2.8 Linear Generalized Additive Model . . . 21
- 2.3 Classification methods . . . 22
  - 2.3.1 Ensemble method . . . 22
  - 2.3.2 Gaussian Naive Bayes . . . 23
  - 2.3.3 Nearest Neighbors Classification . . . 23
  - 2.3.4 Support Vector Machine . . . 23
  - 2.3.5 Decision Tree Classification . . . 24

3 Results and Discussion 25
- 3.1 Regressors . . . 25
  - 3.1.1 Metrics . . . 25
  - 3.1.2 Results of regression methods with 10-fold and raw data set . . . 26

3.1.3 Results of regression methods with 10-fold and modified data set . . . 30
3.1.4 Results of regression methods with "leave one year out" and modified data set . . . 42
3.2 Classifiers . . . 47
3.2.1 Metrics . . . 47
3.2.2 Labels of the data set . . . 48
3.2.3 Results of classification methods . . . 48

4 Conclusion and Future Work 55
4.1 Conclusion . . . 55
4.2 Future Work . . . 55

Appendix A. Figures of the data set 57

Bibliography 69

Chapter 1

Introduction

1.1 Hop

The hop plant is an herbaceous perennial, with an average growth in hop gardens of around 15 years. At the end of March every year, plants are cut, which makes them start rapid growth. They form lateral shoots in June, bloom in July, bear fruit in August and are harvested at the beginning of September[1].

The quality of hops is sensitive to many pests and diseases, which can cause complete loss at worst depending on the situation due to the reduction of quantity or quality of yield. In Europe, hops can be attacked by the larvae of several moths, but only the attack caused by European corn borer (ECB) reaches damage level frequently[2].

1.2 European corn borer (Ostrinia nubilalis)

European corn borer (ECB) is a pest of hundreds of crop and weed species and the earliest report of ECB was in Massachusetts, North America in 1917. The life circle of ECB consists of four stages: egg, larva, pupa, adult. Eggs hatch in four to nine days. The average duration of the larva stage is about 50 days, which is significantly influenced by weather conditions. The duration of the pupa stage is usually about 12 days. The total longevity of adults is normally 18 to 24 days. The moths (ECBs which are in their adult stage) are most active during the first three to four hours of the warm night[3].

There is only a short time interval before the larvae start to bore into the plants, which means that the time where ECB larvae can be exposed to

natural enemies or insecticides is also short. So, the optimal time window of ECB control methods is limited and monitoring the number of ECB is very important[1].

Chapter 2

Materials and Methods

2.1 Data set

The data set used by this thesis consists of weather conditions, the number of ECB captured during the night and the corresponding date, which is collected by the Slovenian Institute of Hop Research and Brewing from 8 pm to 6 am each day from April 30 to September 28, each year from 1999 to 2017. In all, the data set includes 2875 different observations.

The first part is the weather conditions data. It contains temperature, relative humidity and precipitation of each day from 8 pm to 6 am. For each observation there are $3 \times 11 = 33$ different variables listed below:

- *P-n*: The precipitation at measured n o'clock.
- *RH-n*: The relative humidity measured at n o'clock.
- *T-n*: The temperature measured at n o'clock.

The second part is the number of ECBs captured each day from 8 pm to 6 am. For each observation, there is only one variable that is denoted as *No.ECB*. *No.ECB* ranges from 0 to 178 where 1515 observations (52.7%) are 0 and 2186 observations (76.0%) are from 0 to 4.

In this thesis, each weather condition data is considered as an input object and each number of ECBs captured is considered as the desired output value.

2.1.1 Missing values

78 observations are incomplete and 13 observations are missing. The details are as follows.

- May 2, 2001: T-6, RH-6 and P-6 are missing.
- July 17, 2003: No. ECB is missing.
- July 4, 2004: No. ECB is missing.
- July 6, 2004: No. ECB is missing.
- August 20, 2004: No. ECB is missing.
- From July 11 to July 14 in 2006: many weather conditions data is missing.
- From July 18 to September 27 in 2006: many weather conditions data is missing.
- September 28, 2006: no data.
- From September 17 to September 28 in 2011: no data.
- August 31, 2014: P-24 is missing.

2.1.2 Some characters of the data set

Table 2.1 shows some common values of *No.ECB*. Table 2.2 and Table 2.3 show some statistics of the data set.

Value	Count	Frequency(%)
0	1515	52.7%
1	259	9.0%
2	162	5.6%
3	132	4.6%
4	118	4.1%

Table 2.1. Some common values of No. ECB

	Missing	Mean	Variance	Skewness	Kurtosis
No. ECB	4	4.836642	150.8043	5.903181	50.33336
P-20	61	0.152594	1.054112	13.83706	268.7008
P-21	64	0.159409	1.090832	12.91547	239.0446
P-22	61	0.12779	0.631407	10.17535	134.05
P-23	59	0.133913	0.85388	17.52747	492.2715
P-24	63	0.121159	0.474974	8.38143	80.84076
P-1	61	0.123419	0.534734	9.551535	111.203
P-2	62	0.128333	0.588518	9.219088	101.777
P-3	66	0.116785	0.534402	11.83556	193.1219
P-4	63	0.112624	0.473206	10.84216	170.1876
P-5	65	0.116584	0.460964	10.44177	144.634
P-6	63	0.114865	0.440177	10.74581	151.2073
RH-20	27	76.36822	261.0164	-0.41467	-0.76321
RH-21	30	82.18517	206.3853	-0.82392	-0.01186
RH-22	27	85.96763	168.1773	-1.21275	1.150634
RH-23	25	88.32253	141.7709	-1.50505	2.209549
RH-24	28	89.9229	125.5751	-1.73815	3.138128
RH-1	25	91.12674	110.9978	-1.88815	3.796106
RH-2	28	92.02083	99.05657	-2.0103	4.474688
RH-3	32	92.78558	87.34178	-2.1555	5.435425
RH-4	29	93.3193	79.55737	-2.33054	6.653427
RH-5	31	93.56798	71.88097	-2.2873	6.691606
RH-6	29	91.87512	88.75611	-1.70433	3.425312
T-20	27	18.68013	22.17777	-0.02761	-0.30177
T-21	30	17.5477	17.41182	-0.06946	-0.2304
T-22	27	16.81833	15.74325	-0.08795	-0.25637
T-23	25	16.32502	16.22566	0.024442	-0.12865
T-24	28	15.94391	17.87834	0.227462	0.257033
T-1	25	15.59204	19.76438	0.418345	0.694075
T-2	28	15.27201	21.73444	0.587303	1.032981
T-3	32	14.98597	23.50265	0.733274	1.368536
T-4	29	14.66974	24.08997	0.75789	1.462605
T-5	31	14.54625	25.3547	0.839264	1.716777
T-6	29	15.07917	24.6063	0.747846	1.358172

Table 2.2. Some statistics of the data set (1)

	Min	Median	Q3	95-th percentile	Max
No. ECB	0	0	4	23	178
P-20	0	0	0	0.5	28.2
P-21	0	0	0	0.5	27.6
P-22	0	0	0	0.4	17
P-23	0	0	0	0.4	31.4
P-24	0	0	0	0.4	9.8
P-1	0	0	0	0.4	11.2
P-2	0	0	0	0.4	13.2
P-3	0	0	0	0.4	17.4
P-4	0	0	0	0.4	16.8
P-5	0	0	0	0.5	13.6
P-6	0	0	0	0.4	12.2
RH-20	31.3	78.8	90.2	98.8	100
RH-21	32.2	85.5	93.9	99.8	100
RH-22	32	89.6	96	100	100
RH-23	32.9	91.8	97.2	100	100
RH-24	35.3	93.5	98.3	100	100
RH-1	39.6	94.6	98.9	100	100
RH-2	41.1	95.4	99.3	100	100
RH-3	39.1	95.8	99.6	100	100
RH-4	37.7	96.1	99.7	100	100
RH-5	36.6	96.18	99.8	100	100
RH-6	37.9	94.885	99.7	100	100
T-20	3.3	18.65	22	26.6	31.5
T-21	4.5	17.7	20.3	24.5	29
T-22	4.2	16.9	19.5	23.3	28.8
T-23	3.9	16.3	19	23	29.6
T-24	3.1	16	18.5	23.2	32.3
T-1	2.4	15.5	18.075	23.555	33.4
T-2	2.4	15	17.7	24.17	34.6
T-3	1.4	14.7	17.3	24.6	36.6
T-4	1.7	14.4	17	24.575	36.63
T-5	1.8	14.3	17	24.688	37.36
T-6	2.7	14.845	17.6	24.7425	36.92

Table 2.3. Some statistics of the data set (2)

2.2 Regression methods

The following are all examined regression methods.

The version of scikit-learn is 0.22.2[4]. If there is no additional description, all parameters are default, which can be found at https://scikit-learn.org/0.22/modules/classes.html.

The version of pyGAM is 0.8.0[5]. If there is no additional description, all parameters are default, which can be found at https://pygam.readthedocs.io/en/latest/api/api.html

2.2.1 PLS regression

PLSRegression: sklearn.cross_decomposition.PLSRegression

Partial least squares regressor in scikit-learn, which implements PLS2 algorithms or PLS1 algorithms in case of one dimensional response.

Partial least squares regression (PLS regression) was introduced by the Swedish statistician Herman O. A. Wold and is most widely used in chemometrics and its related areas today. It projects the predicted variables and the observable variables to a new space and then finds a linear model of them within the new space[6], which makes it more robust than than classical linear regression[7][8]. Robust means that the model parameters do not change a lot when new observable variables are added.

2.2.2 Ensemble method

AdaBoost

AdaBoostRegressor: sklearn.ensemble.AdaBoostRegressor

An AdaBoost regressor in scikit-learn, which implements AdaBoost.R2 algorithm[9]. Here the decision tree regressor with max_depth=3 is used as the base estimator (weak learner).

AdaBoost (Adaptive Boosting) is introduced by Yoav Freund and Robert Schapire in 1995. It is an ensemble learning method which can be used with other machine learning algorithm to improve performance. This method combines outputs of several "weak learners" into the final output by weighted summation. If the prediction of each weak learner is slightly better than random guessing, the final model can be a strong learner. It trains these "weak learners" one by one. When succeeding weak learner is training, weights of instances that performed poorly in the previous weak learner are increased[10].

Bagging meta-estimator

BaggingRegressor: sklearn.ensemble.BaggingRegressor

A bagging regressor in scikit-learn, in this thesis the default parameters are used, which means that the base estimator is a decision tree.

Bagging (**b**ootstrap **agg**regat**ing**) is an ensemble learning method that was first introduced by Leo Breiman in 1994. The bagging method trains several base estimators on random subsets of the original data set and then combines their outputs (by voting or by averaging) into a final output in order to reduce the variance of the base estimator[11][12][13][14].

Random Forest

RandomForestRegressor: sklearn.ensemble.RandomForestRegressor

A random forest regressor in scikit-learn.

Random forest is an ensemble learning method. The first random decision forests algorithm was created by Tin Kam Ho in 1995 and extended by Leo Breiman and Adele Cutler. A random forest trains a number of decision trees on random subsets of the original data set and takes the average of outputs as the final output. During the construction of decision trees, the best split is found from a random subset of the features based on, e.g., information gain or the Gini impurity when each node is being split[15]. This method reduce the overfitting to the training set, which is a habit of decision trees.

A brief description of decision trees is in section 2.2.7 *Decision Tree* part.

Extremely Randomized Trees

ExtraTreesRegressor: sklearn.ensemble.ExtraTreeRegressor

An extra-trees regressor in scikit-learn.

Extremely randomized trees (extra-trees) is an ensemble learning method. An extra-trees estimator trains a number of extremely randomized trees on the whole original data set and takes the average of outputs as the final output to improve the performance[16]. Compared to random forest method, main differences are following. First, extra-trees uses the whole data set instead of subsets of the data set. Second, it draws random cut-point for each feature when each node is being split and the best of these cut-points is selected.

A brief description of extremely randomized tree is in section 2.2.7 Extremely *Randomized Tree* part.

Gradient Tree Boosting

GradientBoostingRegressor: sklearn.ensemble.GradientBoostingRegressor

A gradient tree boosting regressor in scikit-learn.

Gradient Tree Boosting is also called Gradient Boosted Decision Trees (GBDT). It is an ensemble learning method. This method is similar to AdaBoost, which is described in section 2.2.2 *AdaBoost* part. The difference is that the succeeding weak learner fits the negative gradient instead of weighted data set when it is training[17].

2.2.3 Bayesian Regression

Bayesian regression model is a probabilistic model, where the output y is assumed to be Gaussian distributed around Xw, which is like the following[18]:

$$p(y|X, w, \alpha) = \mathcal{N}(y|Xw, \alpha) \tag{2.1}$$

where, α is a random variable that is to be estimated from the data.

Bayesian Ridge Regression

BayesianRidge: sklearn.linear_model.BayesianRidge

A bayesian ridge regressor in scikit-learn.

Bayesian ridge regression estimates a bayesian regression model where, ω is given by a spherical Gaussian:

$$p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}\mathbf{I}_p) \tag{2.2}$$

the priors over α and λ are gamma distributions and estimated by maximizing the log marginal likelihood during the training of the model[18].

The implementation of scikit-learn is based on the algorithm described in *Sparse Bayesian Learning and the Relevance Vector Machine (Michael E. Tipping, 2001)*[19] where the update of α and λ is done as suggested in *Bayesian Interpolation(David J. C. MacKay, 1992)*[20].

Bayesian ARD regression

ARDRegression: sklearn.linear_model.ARDRegression

A bayesian ARD regressor in scikit-learn.

Bayesian ARD (Automatic Relevance Determination) regression estimates a bayesian regression model where ω is an axis-parallel, elliptical Gaussian distribution[21]:

$$p(w|\lambda) = \mathcal{N}(w|0, A^{-1}) \tag{2.3}$$

with $\mathbf{diag}(A) = \lambda = \{\lambda_1, ..., \lambda_p\}$.

2.2.4 Linear Model

Linear model assumes that the predicted value is a linear combination of the features:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + ... + w_p x_p \tag{2.4}$$

where,

$\hat{y}$ is the predicted value,

w is the vector of model's coefficients $w = (w_1, ..., w_p)$,

Kernel ridge regression

KernelRidge: sklearn.kernel_ridge.KernelRidge

A kernel ridge regressor in scikit-learn.

Kernel ridge regression (KRR) is ridge regression (linear least squares with l2 regularization) with the kernel trick[22].

A ridge model is a linear model, which minimizes the following objective function:

$$\min_w ||Xw - y||_2^2 + \alpha ||w||_2^2 \tag{2.5}$$

where, $\alpha \geq 0$ is complexity parameter.

Kernel trick is a method to transform the nonlinear separable problem of low dimensional space into the linear separable problem of high dimensional space.

Elastic-Net

ElasticNetCV: sklearn.linear_model.ElasticNet

A linear regressor with combined L1 and L2 priors as regularizer in scikit-learn.

ElasticNet is a linear regression model trained with both ℓ_1 and ℓ_2 -norm regularization of the coefficients, which minimizes the following objective function[23]:

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2 \tag{2.6}$$

where, α and l1_ratio ρ are hyper parameters and regressor *ElasticNetCV* selects them by cross-validation.

Least Angle Regression

LarsCV: sklearn.linear_model.LarsCV

A cross-validated Least Angle Regression model in scikit-learn.

Least Angle Regression (LARS) algorithm is developed by Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani. It is similar to classic forward selection procedure and the difference is that LARS procedure proceeds in a direction equiangular among all variables which are already found at each step[24].

Lasso

LassoCV: sklearn.linear_model.LassoCV

A lasso linear model with iterative fitting along a regularization path in scikit-learn and the best model is selected by cross-validation.

Lasso is a linear model with an added regularization term, which minimizes the following objective function[25]:

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha||w||_1 \tag{2.7}$$

where, $\alpha \geq 0$ is complexity parameter.

Passive Aggressive Algorithms

PassiveAggressiveRegressor: sklearn.linear_model.PassiveAggressiveRegressor

A passive aggressive regressor in scikit-learn.

The passive-aggressive algorithms are a family of algorithms for online learning where data is in sequential order and the model is updated at each step. They are similar to the perceptron algorithms. The difference is that they include a regularization parameter and do not require a learning rate[26].

2.2.5 Nearest Neighbors Regression

KNeighborsRegressor: sklearn.neighbors.KNeighborsRegressor

A regressor based on k-nearest neighbors in scikit-learn.

This regression method predict the target by interpolation of the k nearest neighbors in the training set[27].

2.2.6 Support Vector Machine

Epsilon-Support Vector Regression

SVR: sklearn.svm.SVR

A epsilon-support vector regression model in scikit-learn. Default parameters are used, which means it is a svm regressor with rbf kernel.

Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, and a vector $y \in \mathbb{R}^n$ the regressor solve the following problem:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*) \tag{2.8}$$

$$\begin{aligned} \text{subject to: } & y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i, \\ & w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*, \\ & \zeta_i, \zeta_i^* \geq 0, i = 1, ..., n \end{aligned} \tag{2.9}$$

Whose dual problem is:

$$\min_{\alpha,\alpha^*} \frac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \varepsilon e^T (\alpha + \alpha^*) - y^T (\alpha - \alpha^*) \tag{2.10}$$

$$\begin{aligned} \text{subject to: } & e^T (\alpha - \alpha^*) = 0 \\ & 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, ..., n \end{aligned} \tag{2.11}$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel, where function ϕ maps the training vectors into a higher dimensional space.

The decision function is the following:

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + \rho \tag{2.12}$$

where, ρ is a constant[28].

Linear Support Vector Regression

LinearSVR: sklearn.svm.LinearSVR

A linear support vector regression model in scikit-learn. It is a faster implementation than SVR but only considers linear kernels.

Nu Support Vector Regression

NuSVR: sklearn.svm.NuSVR

A nu support vector regression model in scikit-learn.

It is similar to *SVR* but uses a parameter to control the number of support vectors.

2.2.7 Decision Tree Regression

Decision Tree

DecisionTreeRegressor: sklearn.tree.DecisionTreeRegressor

A decision tree regressor in scikit-learn.

Decision Tree is a non-parametric supervised learning method. It predict target variables by learning several simple decision rules from features[29].

Extremely Randomized Tree

ExtraTreeRegressor: sklearn.tree.ExtraTreeRegressor

An extremely randomized tree regressor in scikit-learn.

Extremely randomized tree has the same basic idea as decision tree. The difference is that when the samples of a node is being separated, features are selected randomly[16].

2.2.8 Linear Generalized Additive Model

LinearGAM: pygam.pygam.LinearGAM

A linear GAM in pyGAM.

Generalized additive model can be easily written as the following formula:

$$g(E[y]) = \Sigma f_i(x_i) \tag{2.13}$$

where g is link function, $f_i(x_i)$ shape function[30].

2.3 Classification methods

The following are all examined classification methods.

The version of scikit-learn is 0.22.2[4]. If there is no additional description, all parameters are default, which can be found at https://scikit-learn.org/0.22/modules/classes.html.

The version of pyGAM is 0.8.0[5]. If there is no additional description, all parameters are default, which can be found at https://pygam.readthedocs.io/en/latest/api/api.html

2.3.1 Ensemble method

AdaBoost

AdaBoostClassifier: sklearn.ensemble.AdaBoostClassifier

An AdaBoost classifier in scikit-learn.

A brief description of AdaBoost is in section 2.2.2 *AdaBoost* part.

Bagging meta-estimator

BaggingClassifier: sklearn.ensemble.BaggingClassifier()

A decision tree classifier with bagging in scikit-learn. And decision tree is the base estimator.

A brief description of bagging meta-estimator is in section 2.2.2 *Bagging meta-estimator* part.

A brief description of decision tree is in section 2.2.7 *Decision Tree* part.

Random Forest

RandomForestClassifier: sklearn.ensemble.RandomForestClassifier

A random forest classifier in scikit-learn.

A brief description of random forest is in section 2.2.2 *Random Forest* part.

Extremely Randomized Trees

RandomForestClassifier: sklearn.ensemble.ExtraTreesClassifier

An extra-trees classifier in scikit-learn.

A brief description of extremely randomized trees is in section 2.2.2 *Extremely Randomized Trees* part.

Gradient Tree Boosting

GradientBoostingClassifier: sklearn.ensemble.GradientBoostingClassifier

A gradient boosting classifier in scikit-learn.

A brief description of gradient boosting is in section 2.2.2 *Gradient Tree Boosting* part.

2.3.2 Gaussian Naive Bayes

GaussianNB: sklearn.naive_bayes.GaussianNB

A Gaussian naive bayes classifier in scikit-learn.

Gaussian naive bayes is a naive bayes methods with the following Gaussian likelihood of the features:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.14}$$

where σ_y and μ_y are given by maximum likelihood.

2.3.3 Nearest Neighbors Classification

K_neighbor: sklearn.neighbors.KNeighborsClassifier

A classifier implementing the k-nearest neighbors vote in scikit-learn.

This classification method predict the target based on the k nearest neighbors in the training set.

2.3.4 Support Vector Machine

SVC: sklearn.svm.SVC

A c-support vector classification model in scikit-learn. In this thesis *kernel*='linear', which means that linear kernel is used

Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, in two classes, and a vector $y \in \{1, -1\}^n$, it solves the following problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{2.15}$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0, i = 1, ..., n \tag{2.16}$$

Whose dual is:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{2.17}$$

$$\begin{aligned} &\text{subject to } y^T\alpha = 0 \\ &0 \leq \alpha_i \leq C, i = 1, ..., n \end{aligned} \tag{2.18}$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ and $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the kernel, where function ϕ maps the training vectors into a higher dimensional space.

The decision function is:

$$\text{sgn}(\sum_{i=1}^{n} y_i\alpha_i K(x_i, x) + \rho) \tag{2.19}$$

where ρ is a constant[31].

2.3.5 Decision Tree Classification

Decision Tree

DecisionTreeClassifier: sklearn.tree.DecisionTreeClassifier

A decision tree classifier in scikit-learn.

A brief description of decision tree is in section 2.2.7 *Decision Tree* part.

Extremely Randomized Tree

ExtraTreeClassifier: sklearn.tree.ExtraTreeClassifier

An extremely randomized tree classifier.

A brief description of extremely randomized tree is in section 2.2.7 *Extremely Randomized Tree* part.

Chapter 3

Results and Discussion

In the experimental work different model are evaluated. The process and results of the evaluation are shown in this chapter. Python 3.7.7 with scikit-learn 0.22.2 and pyGAM 0.8.0 on a computer with Windows 10 Pro Version 1909 (64 bits) is used.

3.1 Regressors

3.1.1 Metrics

In order to quantify the quality of predictions, the following metrics are used.

- R^2 score

  R^2 score is also called coefficient of determination. It measures the performance of a model based on the proportion of total variations of the predicted values that can be explained by the regression model[32][33][34].

  The best possible score is 1.0 and it can be negative with the arbitrarily worse model.

  If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, the following is the definition of R^2 score:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.1}$$

  where, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

- Explained variance score

  Explained variance score measures the discrepancy between the predicted values and the true values[35].

  The best possible score is 1.0, lower values means it's a worse model.

  If $\hat{y}_i$ is the predicted value of the i-th sample, y_i is the corresponding true value and is variance, the following is the definition of explained variance score:

$$explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}} \tag{3.2}$$

- Mean squared error

  Mean squared error (MSE) measures the average squared difference between the predicted values and the actual values. Obviously, a good model comes with a small MSE value.

  If $\hat{y}_i$ is the predicted value of the i-th sample, y_i is the corresponding true value and n_{samples} is the number of samples, the following is the definition of MSE:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2 \tag{3.3}$$

3.1.2 Results of regression methods with 10-fold and raw data set

10-fold cross validation

In order to evaluate the performance of models, cross validation technique is used. Although a leave-one-out validation might be the most adequate one among all kinds of cross-validation techniques, it is too time-consuming to use. So 10-fold cross validation is used.

The data set are randomly partitioned into 10 different folds. Each fold contains almost same number of samples from the data set. When regression models are training, one fold is selected as test set and the rest folds are training set. Repeat this procedure 10 times until each of 10 different folds has been selected as test set once and final value of each metrics is given by averaging.

Preprocessing of the data set

Standardization technique is used. Before training each model, mean and variance of training set are computed. And then both training set and test set are standardized with the those mean and variance.

Results of models with the default parameters

To find the most promising regression model, all regressors with the default parameters are evaluated and the results are shown in the Table 3.1.

Table 3.1. Results of regressors with the default parameters, 10-fold and raw data set

Model	R^2	MSE	EV
ElasticNetCV	0.08	133.30	0.09
BayesianRidge	0.08	133.37	0.09
LassoCV	0.08	133.40	0.09
ARDRegression	0.08	133.52	0.08
PLSRegression	0.07	134.73	0.08
LarsCV	0.06	137.18	0.06
ExtraTreesRegressor	-0.01	144.16	0.01
NuSVR	-0.02	148.28	0.03
RandomForestRegressor	-0.03	146.47	-0.02
BaggingRegressor	-0.04	148.35	-0.03
SVR	-0.05	151.25	0.05
GradientBoostingRegressor	-0.07	150.65	-0.06
LinearSVR	-0.08	156.51	0.03
KNeighborsRegressor	-0.09	154.74	-0.08
KernelRidge	-0.11	159.69	0.07
LinearGAM	-0.11	160.24	-0.10
PassiveAggressiveRegressor	-0.47	208.85	-0.44
DecisionTreeRegressor	-1.26	306.08	-1.25
ExtraTreeRegressor	-1.31	317.46	-1.28
AdaBoostRegressor	-2.39	432.39	-0.51

Although all regressors have very bad performances, it is clear that *ElasticNetCV BayesianRidge LassoCV* have the best performance comparing to the rest models given the three metrics mentioned earlier.

Tuning the hyperparameters of *BayesianRidge*

To improve the quality of models, the hyper parameters of the three best model are tuned. Given that the best parameters of *ElasticNetCV* and *LassoCV* are selected by 5-fold cross validation, *BayesianRidge* is the only one that is tuned.

There are four hyperparameters α_1(alpha_1), α_2(alpha_2), λ_1(lambda_1) and λ_2(lambda_2) where are from the gamma prior distributions over α and λ. Given that all these four hyperparameters have the default value 10^{-6}, *BayesianRidge* models with the cross-product of α_1 in [10^{-3},10^{-9}], α_2 in [10^{-3},10^{-9}], λ_1 in [10^{-3},10^{-9}] and λ_2 in [10^{-3},10^{-9}] are tested.

Because it is a huge amount of results (2401 entries), the table of the complete results is not included in this these. 105 sets of hyperparameters (Table 3.2) have the same performances that are better than the rest: $R^2 = 0.08214243$, $explained\_variance = 0.08542188$, $MSE = 133.3707765$ and 21 sets of hyperparameters (Table 3.3) have the same performances that are worse than the rest: $R^2 = 0.082141546$, $explained\_variance = 133.3708157$, $MSE = 0.085420997$.

It is easy to find that small α_1, large λ_1 and small λ_2 make the performance better and larger λ_1 should be tested. But given that hyperparameters doesn't significantly affect the performance, it is time to stop tuning and try something else.

Table 3.2: Best 105 sets of hyperparameters of BayesianRidge

α_1	α_2	λ_1	λ_2	α_1	α_2	λ_1	λ_2
1.00E-05	0.001	0.001	1.00E-07	1.00E-06	0.001	0.001	1.00E-07
1.00E-05	0.001	0.001	1.00E-08	1.00E-06	0.001	0.001	1.00E-08
1.00E-05	0.001	0.001	1.00E-09	1.00E-06	0.001	0.001	1.00E-09
1.00E-05	0.0001	0.001	1.00E-07	1.00E-06	0.0001	0.001	1.00E-07
1.00E-05	0.0001	0.001	1.00E-08	1.00E-06	0.0001	0.001	1.00E-08
1.00E-05	0.0001	0.001	1.00E-09	1.00E-06	0.0001	0.001	1.00E-09
1.00E-05	1.00E-05	0.001	1.00E-07	1.00E-06	1.00E-05	0.001	1.00E-07
1.00E-05	1.00E-05	0.001	1.00E-08	1.00E-06	1.00E-05	0.001	1.00E-08
1.00E-05	1.00E-05	0.001	1.00E-09	1.00E-06	1.00E-05	0.001	1.00E-09
1.00E-05	1.00E-06	0.001	1.00E-07	1.00E-06	1.00E-06	0.001	1.00E-07
1.00E-05	1.00E-06	0.001	1.00E-08	1.00E-06	1.00E-06	0.001	1.00E-08
1.00E-05	1.00E-06	0.001	1.00E-09	1.00E-06	1.00E-06	0.001	1.00E-09
1.00E-05	1.00E-07	0.001	1.00E-07	1.00E-06	1.00E-07	0.001	1.00E-07

1.00E-05	1.00E-07	0.001	1.00E-08	1.00E-06	1.00E-07	0.001	1.00E-08
1.00E-05	1.00E-07	0.001	1.00E-09	1.00E-06	1.00E-07	0.001	1.00E-09
1.00E-05	1.00E-08	0.001	1.00E-07	1.00E-06	1.00E-08	0.001	1.00E-07
1.00E-05	1.00E-08	0.001	1.00E-08	1.00E-06	1.00E-08	0.001	1.00E-08
1.00E-05	1.00E-08	0.001	1.00E-09	1.00E-06	1.00E-08	0.001	1.00E-09
1.00E-05	1.00E-09	0.001	1.00E-07	1.00E-06	1.00E-09	0.001	1.00E-07
1.00E-05	1.00E-09	0.001	1.00E-08	1.00E-06	1.00E-09	0.001	1.00E-08
1.00E-05	1.00E-09	0.001	1.00E-09	1.00E-06	1.00E-09	0.001	1.00E-09
1.00E-07	0.001	0.001	1.00E-07	1.00E-08	0.001	0.001	1.00E-07
1.00E-07	0.001	0.001	1.00E-08	1.00E-08	0.001	0.001	1.00E-08
1.00E-07	0.001	0.001	1.00E-09	1.00E-08	0.001	0.001	1.00E-09
1.00E-07	0.0001	0.001	1.00E-07	1.00E-08	0.0001	0.001	1.00E-07
1.00E-07	0.0001	0.001	1.00E-08	1.00E-08	0.0001	0.001	1.00E-08
1.00E-07	0.0001	0.001	1.00E-09	1.00E-08	0.0001	0.001	1.00E-09
1.00E-07	1.00E-05	0.001	1.00E-07	1.00E-08	1.00E-05	0.001	1.00E-07
1.00E-07	1.00E-05	0.001	1.00E-08	1.00E-08	1.00E-05	0.001	1.00E-08
1.00E-07	1.00E-05	0.001	1.00E-09	1.00E-08	1.00E-05	0.001	1.00E-09
1.00E-07	1.00E-06	0.001	1.00E-07	1.00E-08	1.00E-06	0.001	1.00E-07
1.00E-07	1.00E-06	0.001	1.00E-08	1.00E-08	1.00E-06	0.001	1.00E-08
1.00E-07	1.00E-06	0.001	1.00E-09	1.00E-08	1.00E-06	0.001	1.00E-09
1.00E-07	1.00E-07	0.001	1.00E-07	1.00E-08	1.00E-07	0.001	1.00E-07
1.00E-07	1.00E-07	0.001	1.00E-08	1.00E-08	1.00E-07	0.001	1.00E-08
1.00E-07	1.00E-07	0.001	1.00E-09	1.00E-08	1.00E-07	0.001	1.00E-09
1.00E-07	1.00E-08	0.001	1.00E-07	1.00E-08	1.00E-08	0.001	1.00E-07
1.00E-07	1.00E-08	0.001	1.00E-08	1.00E-08	1.00E-08	0.001	1.00E-08
1.00E-07	1.00E-08	0.001	1.00E-09	1.00E-08	1.00E-08	0.001	1.00E-09
1.00E-07	1.00E-09	0.001	1.00E-07	1.00E-08	1.00E-09	0.001	1.00E-07
1.00E-07	1.00E-09	0.001	1.00E-08	1.00E-08	1.00E-09	0.001	1.00E-08
1.00E-07	1.00E-09	0.001	1.00E-09	1.00E-08	1.00E-09	0.001	1.00E-09
1.00E-09	0.001	0.001	1.00E-07	1.00E-09	1.00E-06	0.001	1.00E-09
1.00E-09	0.001	0.001	1.00E-08	1.00E-09	1.00E-07	0.001	1.00E-07
1.00E-09	0.001	0.001	1.00E-09	1.00E-09	1.00E-07	0.001	1.00E-08
1.00E-09	0.0001	0.001	1.00E-07	1.00E-09	1.00E-07	0.001	1.00E-09
1.00E-09	0.0001	0.001	1.00E-08	1.00E-09	1.00E-08	0.001	1.00E-07
1.00E-09	0.0001	0.001	1.00E-09	1.00E-09	1.00E-08	0.001	1.00E-08
1.00E-09	1.00E-05	0.001	1.00E-07	1.00E-09	1.00E-08	0.001	1.00E-09
1.00E-09	1.00E-05	0.001	1.00E-08	1.00E-09	1.00E-09	0.001	1.00E-07
1.00E-09	1.00E-05	0.001	1.00E-09	1.00E-09	1.00E-09	0.001	1.00E-08
1.00E-09	1.00E-06	0.001	1.00E-07	1.00E-09	1.00E-09	0.001	1.00E-09

1.00E-09	1.00E-06	0.001	1.00E-08				

Table 3.3: Worst 21 sets of hyperparameters of BayesianRidge

α_1	α_2	λ_1	λ_2
0.001	0.001	1.00E-07	0.001
0.001	0.001	1.00E-08	0.001
0.001	0.001	1.00E-09	0.001
0.001	0.0001	1.00E-07	0.001
0.001	0.0001	1.00E-08	0.001
0.001	0.0001	1.00E-09	0.001
0.001	1.00E-05	1.00E-07	0.001
0.001	1.00E-05	1.00E-08	0.001
0.001	1.00E-05	1.00E-09	0.001
0.001	1.00E-06	1.00E-07	0.001
0.001	1.00E-06	1.00E-08	0.001
0.001	1.00E-06	1.00E-09	0.001
0.001	1.00E-07	1.00E-07	0.001
0.001	1.00E-07	1.00E-08	0.001
0.001	1.00E-07	1.00E-09	0.001
0.001	1.00E-08	1.00E-07	0.001
0.001	1.00E-08	1.00E-08	0.001
0.001	1.00E-08	1.00E-09	0.001
0.001	1.00E-09	1.00E-07	0.001
0.001	1.00E-09	1.00E-08	0.001
0.001	1.00E-09	1.00E-09	0.001

3.1.3 Results of regression methods with 10-fold and modified data set

Modifications of the data set

Given that all models that used previously don't learn from the history of the weather condition, it is necessary to add historical weather conditions information during the model fitting.

To add such information, four different new data sets are created.

The first data set (denoted as *f.s. 1*) includes the following information: daily mean, daily maximum, daily minimum, daily variance, daily median,

mean of the day and the past two days, maximum of the day and the past two days, minimum of the day and the past two days,mean of the day and the past six days, maximum of the day and the past six days, minimum of the day and the past six days of each variable for each day in the data set.

The second data set (denoted as *f.s. 13*) is similar to *f.s. 1*, which includes all information of *f.s. 1* and mean of the day and the past twelve days, maximum of the day and the past twelve days, minimum of the day and the past twelve days of each variable for each day in the data set.

The third data set (denoted as *f.s. 2*) consists of the differences between the raw data and the mean of the day and the past six days of each variable for each day. In short, it is obtained by subtracting the seven-day average of the corresponding variable.

The fourth data set (denoted as *f.s. 3*) consists of the differences between the raw data and the corresponding variable of each past seven days.

Preprocessing of the data set

Standardization technique is also used in this time. Both training on raw data AND new created data (denoted as the name of the new created data set) and training on ONLY new created data (denoted as the name of the new created data set + ONLY) are tested.

Results of models with the default parameters

All regressors with the default parameters are evaluated in the same way as section 3.1.2 by using modified data set and the results are shown in the Table 3.4.

ExtraTreesRegressor and *RandomForestRegress* have a significant performance improvement with the help of data set *f.s. 13*.

Table 3.4: Results of models with the default parameters, 10-fold and modified data set

Model	Dataset	R^2	MSE	EV
ExtraTreesRegressor	f.s. 13 ONLY	0.46	65.16	0.47
RandomForestRegressor	f.s. 13 ONLY	0.39	73.41	0.40
ExtraTreesRegressor	f.s. 13	0.36	78.65	0.37
RandomForestRegressor	f.s. 13	0.31	84.04	0.32
GradientBoostingRegressor	f.s. 13 ONLY	0.29	83.71	0.29
RandomForestRegressor	f.s. 1 ONLY	0.28	119.83	0.29

ExtraTreesRegressor	f.s. 1 ONLY	0.27	118.20	0.28
GradientBoostingRegressor	f.s. 13	0.22	92.30	0.23
RandomForestRegressor	f.s. 1	0.21	129.22	0.23
GradientBoostingRegressor	f.s. 1 ONLY	0.21	133.68	0.21
BaggingRegressor	f.s. 13 ONLY	0.20	100.71	0.21
ExtraTreesRegressor	f.s. 1	0.18	133.77	0.19
GradientBoostingRegressor	f.s. 1	0.18	134.22	0.20
BaggingRegressor	f.s. 13	0.12	108.55	0.13
BayesianRidge	f.s. 13	0.11	113.97	0.11
ElasticNetCV	f.s. 13	0.10	113.98	0.11
LassoCV	f.s. 13	0.10	114.08	0.11
BayesianRidge	f.s. 13 ONLY	0.10	114.75	0.11
ARDRegression	f.s. 13	0.09	114.97	0.10
ElasticNetCV	f.s. 13 ONLY	0.09	115.05	0.10
ARDRegression	f.s. 13 ONLY	0.09	115.48	0.10
LassoCV	f.s. 13 ONLY	0.09	115.48	0.10
ElasticNetCV	f.s. 1	0.09	116.32	0.10
LarsCV	f.s. 13 ONLY	0.09	150.69	0.10
LassoCV	f.s. 1	0.09	150.78	0.10
BayesianRidge	f.s. 1 ONLY	0.09	150.79	0.10
BayesianRidge	f.s. 1	0.09	151.18	0.10
ElasticNetCV	f.s. 1 ONLY	0.09	151.24	0.10
ARDRegression	f.s. 1	0.09	151.28	0.10
LassoCV	f.s. 1 ONLY	0.09	151.43	0.10
BaggingRegressor	f.s. 1	0.08	133.30	0.09
ElasticNetCV	raw feat.	0.08	133.37	0.09
ARDRegression	f.s. 3	0.08	133.40	0.09
BayesianRidge	raw feat.	0.08	133.52	0.08
ARDRegression	f.s. 1 ONLY	0.08	151.57	0.09
LassoCV	raw feat.	0.08	153.11	0.09
LassoCV	f.s. 3	0.08	154.28	0.10
ARDRegression	raw feat.	0.08	172.83	0.09
ElasticNetCV	f.s. 3	0.08	173.64	0.09
LarsCV	f.s. 1 ONLY	0.08	173.92	0.08
PLSRegression	f.s. 13 ONLY	0.07	117.98	0.09
PLSRegression	raw feat.	0.07	118.40	0.08
PLSRegression	f.s. 13	0.07	134.73	0.08
BayesianRidge	f.s. 3	0.07	155.25	0.08
PLSRegression	f.s. 1 ONLY	0.07	155.34	0.08

PLSRegression	f.s. 1	0.07	175.72	0.07
BaggingRegressor	f.s. 1 ONLY	0.06	120.33	0.07
LarsCV	f.s. 13	0.06	137.18	0.06
LarsCV	raw feat.	0.06	148.57	0.08
NuSVR	f.s. 13 ONLY	0.05	123.18	0.09
PLSRegression	f.s. 3	0.05	123.73	0.08
NuSVR	f.s. 13	0.05	178.94	0.06
KNeighborsRegressor	f.s. 1 ONLY	0.04	153.75	0.05
SVR	f.s. 13 ONLY	0.03	125.27	0.10
LarsCV	f.s. 1	0.03	126.03	0.09
SVR	f.s. 13	0.03	161.45	0.04
ExtraTreesRegressor	f.s. 3	0.02	165.89	0.06
BaggingRegressor	f.s. 3	0.02	166.37	0.06
NuSVR	f.s. 1 ONLY	0.02	182.60	0.03
NuSVR	f.s. 1	0.02	184.24	0.03
LinearGAM	f.s. 1 ONLY	0.01	157.21	0.02
SVR	f.s. 1 ONLY	0.01	168.10	0.08
SVR	f.s. 1	0.00	168.50	0.07
GradientBoostingRegressor	f.s. 3	0.00	188.29	0.00
LarsCV	f.s. 3	0.00	188.93	0.00
LarsCV	f.s. 3 ONLY	0.00	189.76	0.00
BayesianRidge	f.s. 3 ONLY	0.00	189.77	0.00
ElasticNetCV	f.s. 3 ONLY	-0.01	144.16	0.01
LassoCV	f.s. 3 ONLY	-0.01	159.83	0.00
PLSRegression	f.s. 3 ONLY	-0.01	167.21	0.00
LassoCV	f.s. 2	-0.01	167.31	0.01
LarsCV	f.s. 2	-0.01	167.34	0.00
ElasticNetCV	f.s. 2	-0.01	167.35	0.00
BayesianRidge	f.s. 2	-0.01	167.40	0.00
KNeighborsRegressor	f.s. 1	-0.01	167.41	0.00
ARDRegression	f.s. 2	-0.01	189.90	0.00
ExtraTreesRegressor	raw feat.	-0.01	189.90	0.00
PLSRegression	f.s. 2	-0.01	189.98	0.00
NuSVR	f.s. 3	-0.01	190.66	-0.01
ARDRegression	f.s. 3 ONLY	-0.01	192.13	0.04
NuSVR	raw feat.	-0.02	148.28	0.03
LinearSVR	f.s. 13	-0.03	123.39	-0.02
RandomForestRegressor	raw feat.	-0.03	132.50	0.06
LinearGAM	f.s. 13 ONLY	-0.03	133.33	0.05

RandomForestRegressor	f.s. 3	-0.03	146.47	-0.02
LinearSVR	f.s. 3	-0.03	194.27	-0.03
LinearSVR	f.s. 13 ONLY	-0.03	196.04	0.04
BaggingRegressor	raw feat.	-0.04	128.94	-0.03
SVR	f.s. 3	-0.04	148.35	-0.03
LinearSVR	f.s. 1	-0.04	169.49	-0.01
KNeighborsRegressor	f.s. 13	-0.04	174.72	0.05
ExtraTreesRegressor	f.s. 2	-0.04	175.14	0.04
LinearSVR	f.s. 1 ONLY	-0.04	175.48	0.00
NuSVR	f.s. 2	-0.04	196.30	0.05
SVR	raw feat.	-0.05	126.25	-0.04
NuSVR	f.s. 3 ONLY	-0.05	151.25	0.05
KNeighborsRegressor	f.s. 13 ONLY	-0.05	198.84	0.00
RandomForestRegressor	f.s. 2	-0.06	170.19	-0.03
GradientBoostingRegressor	raw feat.	-0.07	150.65	-0.06
GradientBoostingRegressor	f.s. 2	-0.08	176.74	-0.06
LinearSVR	raw feat.	-0.09	154.74	-0.08
KNeighborsRegressor	raw feat.	-0.09	156.51	0.03
BaggingRegressor	f.s. 3 ONLY	-0.09	178.21	0.09
KernelRidge	f.s. 1 ONLY	-0.09	203.60	-0.08
ExtraTreesRegressor	f.s. 3 ONLY	-0.09	204.20	-0.08
KernelRidge	f.s. 1	-0.10	139.04	0.09
KernelRidge	f.s. 13 ONLY	-0.10	178.69	0.09
SVR	f.s. 2	-0.11	159.69	0.07
SVR	f.s. 3 ONLY	-0.11	160.24	-0.10
KNeighborsRegressor	f.s. 3	-0.11	179.97	-0.10
LinearGAM	raw feat.	-0.11	184.65	0.01
BaggingRegressor	f.s. 2	-0.11	204.32	-0.09
KernelRidge	raw feat.	-0.11	208.67	0.00
GradientBoostingRegressor	f.s. 3 ONLY	-0.12	140.33	0.09
KernelRidge	f.s. 13	-0.12	187.19	0.00
LinearSVR	f.s. 2	-0.12	208.08	-0.11
LinearSVR	f.s. 3 ONLY	-0.13	210.46	0.03
KernelRidge	f.s. 3	-0.13	211.98	-0.01
KNeighborsRegressor	f.s. 2	-0.15	185.21	-0.14
RandomForestRegressor	f.s. 3 ONLY	-0.16	215.07	-0.14
KernelRidge	f.s. 2	-0.17	194.05	0.00
KNeighborsRegressor	f.s. 3 ONLY	-0.22	221.86	-0.19
KernelRidge	f.s. 3 ONLY	-0.26	234.13	-0.10

LinearGAM	f.s. 2	-0.31	207.75	-0.30
LinearGAM	f.s. 3	-0.36	166.69	-0.34
PassiveAggressiveRegressor	f.s. 13 ONLY	-0.36	245.76	-0.35
LinearGAM	f.s. 1	-0.39	201.67	-0.38
PassiveAggressiveRegressor	f.s. 1 ONLY	-0.39	226.96	-0.36
DecisionTreeRegressor	f.s. 13	-0.45	158.56	-0.44
LinearGAM	f.s. 3 ONLY	-0.46	263.86	-0.46
PassiveAggressiveRegressor	raw feat.	-0.47	208.85	-0.44
PassiveAggressiveRegressor	f.s. 2	-0.48	233.09	-0.45
LinearGAM	f.s. 13	-0.60	174.41	-0.59
DecisionTreeRegressor	f.s. 13 ONLY	-0.63	186.24	-0.61
ExtraTreeRegressor	f.s. 13	-0.71	197.25	-0.70
DecisionTreeRegressor	f.s. 1	-0.75	243.24	-0.73
DecisionTreeRegressor	f.s. 1 ONLY	-0.75	254.82	-0.73
PassiveAggressiveRegressor	f.s. 1	-0.77	272.64	-0.75
PassiveAggressiveRegressor	f.s. 13	-0.90	232.11	-0.87
DecisionTreeRegressor	f.s. 3	-0.94	353.24	-0.93
ExtraTreeRegressor	f.s. 1 ONLY	-1.05	284.50	-1.03
ExtraTreeRegressor	f.s. 1	-1.14	290.68	-1.11
ExtraTreeRegressor	f.s. 3	-1.15	367.41	-1.14
AdaBoostRegressor	f.s. 3	-1.17	373.03	0.01
ExtraTreeRegressor	f.s. 13 ONLY	-1.24	239.58	-1.22
DecisionTreeRegressor	raw feat.	-1.26	306.08	-1.25
ExtraTreeRegressor	f.s. 3 ONLY	-1.28	398.04	-1.26
ExtraTreeRegressor	f.s. 2	-1.30	335.47	-1.27
ExtraTreeRegressor	raw feat.	-1.31	317.46	-1.28
AdaBoostRegressor	f.s. 13	-1.42	279.10	-0.01
AdaBoostRegressor	f.s. 1	-1.44	337.72	-0.04
DecisionTreeRegressor	f.s. 3 ONLY	-1.49	428.54	-1.46
DecisionTreeRegressor	f.s. 2	-1.50	320.56	-1.46
AdaBoostRegressor	f.s. 3 ONLY	-1.53	442.09	-0.12
AdaBoostRegressor	f.s. 13 ONLY	-1.59	284.01	-0.11
PassiveAggressiveRegressor	f.s. 3	-1.67	472.89	-1.64
AdaBoostRegressor	f.s. 1 ONLY	-1.76	373.36	-0.19
PassiveAggressiveRegressor	f.s. 3 ONLY	-2.26	583.57	-2.19
AdaBoostRegressor	raw feat.	-2.39	432.39	-0.51
AdaBoostRegressor	f.s. 2	-4.03	589.25	-0.50

Tuning the hyperparameters of *ExtraTreesRegressor* and *RandomForestRegress*

To improve the quality of models, the hyper parameters of *ExtraTreesRegressor* and *RandomForestRegress* are tuned.

The following six hyperparameters are going to be tuned for each model:

n_estimators: the number of trees in the forest. max_depth: the maximum depth of the tree. min_samples_split: the minimum number of samples required to split an internal node. max_features: the number of features to consider when looking for the best split.

First, n_estimators is tuned. The results are shown in Table 3.5 and Table 3.6.

Table 3.5: Results of *ExtraTreesRegressor* with different *n_estimators*

n_estimators	R^2	MSE	EV
90	0.461	64.889	0.470
80	0.458	65.057	0.467
140	0.457	65.096	0.466
100	0.457	65.160	0.466
50	0.457	65.475	0.466
110	0.456	65.276	0.465
120	0.456	65.376	0.465
130	0.455	65.366	0.464
150	0.454	65.340	0.464
70	0.453	65.919	0.462
60	0.451	66.027	0.461
40	0.447	66.284	0.456
30	0.438	66.458	0.448
20	0.403	70.424	0.412
10	0.365	74.201	0.375

Table 3.6: Results of *RandomForestRegressor* with different *n_estimators*

n_estimators	R^2	MSE	EV
140	0.396	72.524	0.408
120	0.395	72.572	0.407
150	0.394	72.684	0.406

130	0.394	72.796	0.406
110	0.391	72.980	0.403
100	0.390	73.409	0.401
90	0.387	73.757	0.398
80	0.387	74.030	0.398
60	0.387	74.295	0.399
70	0.387	74.330	0.398
50	0.381	74.611	0.393
40	0.370	75.918	0.382
30	0.351	78.167	0.363
20	0.329	81.225	0.340
10	0.296	84.106	0.307

n_estimators of *ExtraTreesRegressor* and *RandomForestRegressor* are chosen as 90 and 140 respectively.

Then, *max_depth* and *min_samples_split* are tuned. The results are shown in Table 3.7 Table 3.8.

Table 3.7: Results of *ExtraTreesRegressor* with different *max_depth* and *min_samples_split*

max_depth	min_samples_split	R^2	MSE	EV
30	4	0.464	64.980	0.472
50	4	0.463	65.065	0.472
70	4	0.463	65.065	0.472
90	4	0.463	65.065	0.472
None	4	0.463	65.065	0.472
50	2	0.461	64.889	0.470
70	2	0.461	64.889	0.470
90	2	0.461	64.889	0.470
None	2	0.461	64.889	0.470
30	2	0.460	65.019	0.469
50	6	0.447	67.083	0.455
70	6	0.447	67.083	0.455
90	6	0.447	67.083	0.455
None	6	0.447	67.083	0.455
30	6	0.446	67.148	0.455
50	8	0.429	70.157	0.438
70	8	0.429	70.157	0.438
90	8	0.429	70.157	0.438

None	8	0.429	70.157	0.438
10	2	0.428	69.683	0.435
30	8	0.428	70.220	0.437
50	10	0.416	71.968	0.424
70	10	0.416	71.968	0.424
90	10	0.416	71.968	0.424
None	10	0.416	71.968	0.424
30	10	0.416	71.969	0.424
10	6	0.414	72.063	0.420
10	4	0.413	72.546	0.419
30	14	0.412	72.889	0.419
50	14	0.412	72.889	0.419
70	14	0.412	72.889	0.419
90	14	0.412	72.889	0.419
None	14	0.412	72.889	0.419
30	12	0.409	73.258	0.417
50	12	0.409	73.274	0.417
70	12	0.409	73.274	0.417
90	12	0.409	73.274	0.417
None	12	0.409	73.274	0.417
50	16	0.397	74.972	0.405
70	16	0.397	74.972	0.405
90	16	0.397	74.972	0.405
None	16	0.397	74.972	0.405
30	16	0.397	75.031	0.404
10	8	0.392	74.544	0.399
10	10	0.388	76.216	0.395
30	18	0.383	76.859	0.390
50	18	0.383	76.859	0.390
70	18	0.383	76.859	0.390
90	18	0.383	76.859	0.390
None	18	0.383	76.859	0.390
10	12	0.379	77.531	0.386
10	14	0.370	78.625	0.377
10	16	0.356	80.113	0.363
10	18	0.350	80.678	0.357

Table 3.8: Results of *RandomForestRegressor* with different *max_depth* and *min_samples_split*

max_depth	min_samples_split	R^2	MSE	EV
30	4	0.397	72.749	0.409
50	4	0.397	72.773	0.408
70	4	0.397	72.773	0.408
90	4	0.397	72.773	0.408
None	4	0.397	72.773	0.408
30	2	0.396	72.506	0.408
50	2	0.396	72.524	0.408
70	2	0.396	72.524	0.408
90	2	0.396	72.524	0.408
None	2	0.396	72.524	0.408
10	2	0.390	73.387	0.399
50	6	0.388	74.170	0.400
70	6	0.388	74.170	0.400
90	6	0.388	74.170	0.400
None	6	0.388	74.170	0.400
30	6	0.387	74.209	0.400
50	8	0.381	74.956	0.394
70	8	0.381	74.956	0.394
90	8	0.381	74.956	0.394
None	8	0.381	74.956	0.394
30	8	0.381	75.007	0.393
10	4	0.379	75.030	0.388
50	10	0.371	76.276	0.384
70	10	0.371	76.276	0.384
90	10	0.371	76.276	0.384
None	10	0.371	76.276	0.384
30	10	0.371	76.277	0.384
10	6	0.368	76.180	0.378
30	12	0.367	76.723	0.380
50	12	0.367	76.723	0.380
70	12	0.367	76.723	0.380
90	12	0.367	76.723	0.380
None	12	0.367	76.723	0.380
10	8	0.363	77.009	0.373
10	10	0.361	77.596	0.371

30	14	0.361	78.069	0.374
50	14	0.361	78.079	0.374
70	14	0.361	78.079	0.374
90	14	0.361	78.079	0.374
None	14	0.361	78.079	0.374
10	12	0.356	78.269	0.366
30	16	0.356	78.974	0.368
50	16	0.355	78.990	0.368
70	16	0.355	78.990	0.368
90	16	0.355	78.990	0.368
None	16	0.355	78.990	0.368
50	18	0.350	79.615	0.362
70	18	0.350	79.615	0.362
90	18	0.350	79.615	0.362
None	18	0.350	79.615	0.362
30	18	0.350	79.631	0.362
10	14	0.348	79.434	0.359
10	16	0.342	80.511	0.352
10	18	0.336	81.241	0.347

So *max_depth* and *min_samples_split* of *ExtraTreesRegressor* are chosen as 30 and 4 respectively. For *RandomForestRegressor* the values are 30 and 2 respectively, given that its MSE is 0.2 less than the one whose *max_depth* is 30 and *min_samples_split* is 4.

Finally *max_features* are tuned. The results are shown in Table 3.9 Table 3.10.

Table 3.9: Results of *ExtraTreesRegressor* with different *max_features*

max_features	R^2	MSE	EV
42	0.464	64.980	0.472
40	0.457	66.454	0.466
32	0.451	66.261	0.460
28	0.450	67.417	0.459
34	0.443	67.631	0.452
36	0.442	67.885	0.451
30	0.442	68.267	0.450
38	0.438	68.834	0.447
26	0.433	68.839	0.442

20	0.428	70.280	0.437
22	0.425	70.158	0.435
24	0.421	71.438	0.430
18	0.418	72.208	0.427
16	0.413	71.533	0.422
14	0.405	73.800	0.414
12	0.402	74.522	0.410
10	0.384	76.153	0.393
8	0.378	77.376	0.387
6	0.349	81.547	0.359
4	0.342	82.844	0.350
2	0.310	87.499	0.318

Table 3.10: Results of *RandomForestRegressor* with different *max_features*

max_features	R^2	MSE	EV
26	0.418	71.017	0.429
24	0.417	70.906	0.428
18	0.412	72.238	0.423
22	0.411	71.722	0.423
20	0.411	71.737	0.422
16	0.411	71.750	0.422
30	0.409	71.820	0.420
28	0.408	71.788	0.420
14	0.408	72.961	0.420
40	0.406	71.320	0.418
36	0.405	71.819	0.417
34	0.400	72.279	0.412
32	0.400	72.475	0.411
38	0.399	72.450	0.411
12	0.399	73.917	0.410
42	0.396	72.506	0.408
10	0.394	73.754	0.406
8	0.383	75.819	0.394
6	0.382	76.684	0.394
4	0.368	78.592	0.377
2	0.317	86.607	0.326

The *max_features* of *ExtraTreesRegressor* and *RandomForestRegressor* is chosen as 42 and 26 respectively.

Comparing to the situation of no historical information added, the performances are improved, but still not good enough

3.1.4 Results of regression methods with "leave one year out" and modified data set

By adding the historical weather conditions information, we succeed in improving the quality of the prediction. But in the reality, the weather conditions information is collected in chronological order, which means that 10-fold cross validation technique is likely to fail to reflect the actual performance of the model. In short, 10-fold shuffles all the data and probably makes the prediction easier.

"leave one year out"

To reflect the reality, a new way (denoted as leave one year out) to split the training set and the test set for each fold is used.

The data set are partitioned into 19 different folds according to the year of the date, which means that the data of the same year are in the same fold. Then everything is same as the 10-fold cross validation, one fold is selected as test set and the rest folds are training set, except for the fold of 2006 can not be selected as test set, which is because of an amount of missing values.

Results of models with the default parameters

All regressors with the default parameters are evaluated in the same way as section 3.1.3 by using the "leave one year out" cross validation technique and the results are shown in the Table 3.11.

Comparing to the results evaluated by 10-fold (Table 3.4), the performances are obviously worse.

Table 3.11: Results of models with the default parameters, "leave one year out" and modified data set

Model	Dataset	R^2	MSE	EV
NuSVR	f.s. 1 ONLY	0.04	164.67	0.12
NuSVR	f.s. 13 ONLY	0.04	168.55	0.13
NuSVR	f.s. 13	0.04	169.11	0.13

NuSVR	f.s. 1	0.03	165.30	0.11
SVR	f.s. 13 ONLY	0.03	172.40	0.14
SVR	f.s. 1 ONLY	0.02	168.32	0.13
SVR	f.s. 13	0.02	172.54	0.13
SVR	f.s. 1	0.01	168.58	0.13
NuSVR	raw feat.	0.00	161.91	0.08
NuSVR	f.s. 3	0.00	167.03	0.08
SVR	raw feat.	-0.02	164.76	0.10
SVR	f.s. 3	-0.03	171.09	0.09
LinearSVR	f.s. 13	-0.04	178.13	0.09
LinearSVR	f.s. 13 ONLY	-0.04	178.45	0.09
LinearSVR	f.s. 1	-0.05	172.43	0.08
LinearSVR	f.s. 1 ONLY	-0.05	172.53	0.08
LinearSVR	f.s. 3	-0.06	171.75	0.06
LinearSVR	raw feat.	-0.09	169.73	0.06
NuSVR	f.s. 2	-0.09	173.31	0.01
NuSVR	f.s. 3 ONLY	-0.09	173.49	0.00
SVR	f.s. 3 ONLY	-0.17	182.20	0.01
SVR	f.s. 2	-0.17	182.84	0.02
LinearSVR	f.s. 3 ONLY	-0.21	184.60	-0.01
LarsCV	raw feat.	-0.22	151.84	0.06
LinearSVR	f.s. 2	-0.22	185.11	0.00
LarsCV	f.s. 1	-0.23	155.12	0.06
BayesianRidge	raw feat.	-0.28	148.73	-0.03
BayesianRidge	f.s. 3	-0.28	152.72	-0.01
ElasticNetCV	f.s. 3	-0.29	151.37	-0.03
ARDRegression	raw feat.	-0.30	148.58	-0.05
ElasticNetCV	raw feat.	-0.30	148.67	-0.04
LassoCV	raw feat.	-0.30	148.68	-0.04
LassoCV	f.s. 3	-0.30	150.72	-0.03
PLSRegression	raw feat.	-0.32	150.72	-0.03
ElasticNetCV	f.s. 3 ONLY	-0.32	165.12	0.00
BayesianRidge	f.s. 3 ONLY	-0.32	165.17	0.00
LarsCV	f.s. 3 ONLY	-0.32	165.23	0.00
LassoCV	f.s. 3 ONLY	-0.32	165.29	0.00
BayesianRidge	f.s. 1	-0.33	151.89	-0.07
BayesianRidge	f.s. 2	-0.34	164.91	0.00
LarsCV	f.s. 1 ONLY	-0.35	153.31	-0.06
PLSRegression	f.s. 3 ONLY	-0.35	164.76	-0.02

LarsCV	f.s. 2	-0.35	165.28	0.00
ARDRegression	f.s. 1 ONLY	-0.36	150.96	-0.09
BayesianRidge	f.s. 1 ONLY	-0.36	151.78	-0.07
PLSRegression	f.s. 1	-0.36	155.38	-0.04
ARDRegression	f.s. 3 ONLY	-0.36	164.70	-0.03
LassoCV	f.s. 1 ONLY	-0.37	151.40	-0.10
ElasticNetCV	f.s. 1 ONLY	-0.37	151.46	-0.10
ElasticNetCV	f.s. 1	-0.37	152.07	-0.11
PLSRegression	f.s. 1 ONLY	-0.37	155.43	-0.04
PLSRegression	f.s. 2	-0.37	165.05	-0.01
ElasticNetCV	f.s. 2	-0.37	166.35	-0.01
LassoCV	f.s. 2	-0.37	166.35	-0.01
ARDRegression	f.s. 2	-0.37	166.94	-0.01
LassoCV	f.s. 1	-0.38	152.10	-0.11
ARDRegression	f.s. 1	-0.38	152.23	-0.13
PLSRegression	f.s. 3	-0.38	157.90	-0.05
BayesianRidge	f.s. 13	-0.39	155.66	-0.11
LarsCV	f.s. 13 ONLY	-0.40	158.33	-0.08
PLSRegression	f.s. 13	-0.40	160.27	-0.04
ARDRegression	f.s. 3	-0.41	151.42	-0.14
ARDRegression	f.s. 13 ONLY	-0.41	155.01	-0.13
BayesianRidge	f.s. 13 ONLY	-0.41	155.73	-0.10
PLSRegression	f.s. 13 ONLY	-0.41	160.29	-0.05
ElasticNetCV	f.s. 13	-0.42	155.96	-0.14
LassoCV	f.s. 13 ONLY	-0.43	155.90	-0.13
ElasticNetCV	f.s. 13 ONLY	-0.43	155.95	-0.13
LassoCV	f.s. 13	-0.45	156.30	-0.16
KernelRidge	f.s. 2	-0.45	196.34	-0.07
KernelRidge	raw feat.	-0.46	175.43	-0.09
ARDRegression	f.s. 13	-0.47	156.94	-0.18
KernelRidge	f.s. 1 ONLY	-0.50	177.65	-0.14
KernelRidge	f.s. 3 ONLY	-0.57	195.61	-0.26
KernelRidge	f.s. 13 ONLY	-0.58	184.11	-0.17
KernelRidge	f.s. 1	-0.62	181.88	-0.21
GradientBoostingRegressor	f.s. 2	-0.64	177.09	-0.33
LarsCV	f.s. 3	-0.65	174.23	-0.33
GradientBoostingRegressor	raw feat.	-0.66	163.68	-0.40
GradientBoostingRegressor	f.s. 13	-0.68	178.20	-0.44
GradientBoostingRegressor	f.s. 1	-0.71	172.58	-0.45

KernelRidge	f.s. 13	-0.72	188.93	-0.27
BaggingRegressor	f.s. 3	-0.79	167.87	-0.49
BaggingRegressor	raw feat.	-0.80	164.59	-0.54
BaggingRegressor	f.s. 13	-0.83	171.02	-0.50
RandomForestRegressor	raw feat.	-0.84	160.52	-0.48
BaggingRegressor	f.s. 2	-0.87	180.88	-0.53
RandomForestRegressor	f.s. 1	-0.89	171.48	-0.43
RandomForestRegressor	f.s. 2	-0.89	179.21	-0.40
RandomForestRegressor	f.s. 13	-0.89	179.70	-0.44
ExtraTreesRegressor	f.s. 2	-0.90	175.00	-0.39
BaggingRegressor	f.s. 13 ONLY	-0.90	177.82	-0.58
GradientBoostingRegressor	f.s. 13 ONLY	-0.91	181.07	-0.57
ExtraTreesRegressor	f.s. 1	-0.92	166.80	-0.46
ExtraTreesRegressor	raw feat.	-0.96	164.03	-0.56
ExtraTreesRegressor	f.s. 13	-0.96	175.46	-0.48
GradientBoostingRegressor	f.s. 1 ONLY	-0.96	179.59	-0.63
BaggingRegressor	f.s. 3 ONLY	-0.96	181.85	-0.57
ExtraTreesRegressor	f.s. 3	-0.97	166.07	-0.48
GradientBoostingRegressor	f.s. 3	-0.97	173.06	-0.63
ExtraTreesRegressor	f.s. 1 ONLY	-1.00	169.01	-0.51
BaggingRegressor	f.s. 1	-1.01	168.55	-0.67
KNeighborsRegressor	raw feat.	-1.01	173.81	-0.78
KernelRidge	f.s. 3	-1.02	191.34	-0.62
BaggingRegressor	f.s. 1 ONLY	-1.03	167.93	-0.69
RandomForestRegressor	f.s. 1 ONLY	-1.04	175.70	-0.53
RandomForestRegressor	f.s. 13 ONLY	-1.06	186.85	-0.51
GradientBoostingRegressor	f.s. 3 ONLY	-1.07	183.02	-0.65
ExtraTreesRegressor	f.s. 13 ONLY	-1.09	180.62	-0.55
ExtraTreesRegressor	f.s. 3 ONLY	-1.12	181.55	-0.46
RandomForestRegressor	f.s. 3	-1.22	177.49	-0.69
KNeighborsRegressor	f.s. 2	-1.26	187.45	-0.93
PassiveAggressiveRegressor	raw feat.	-1.51	179.75	-1.22
RandomForestRegressor	f.s. 3 ONLY	-1.52	193.81	-0.74
LinearGAM	f.s. 1 ONLY	-1.59	192.01	-1.31
KNeighborsRegressor	f.s. 3	-1.62	187.48	-0.99
KNeighborsRegressor	f.s. 1	-1.70	188.71	-1.31
LarsCV	f.s. 13	-1.74	241.31	-1.43
KNeighborsRegressor	f.s. 1 ONLY	-1.83	191.18	-1.43
KNeighborsRegressor	f.s. 3 ONLY	-1.85	196.84	-1.14

KNeighborsRegressor	f.s. 13	-1.89	197.05	-1.48
KNeighborsRegressor	f.s. 13 ONLY	-2.28	210.11	-1.80
LinearGAM	f.s. 2	-2.28	215.80	-1.94
PassiveAggressiveRegressor	f.s. 1 ONLY	-2.42	210.06	-1.99
PassiveAggressiveRegressor	f.s. 2	-2.57	244.89	-2.10
LinearGAM	raw feat.	-2.63	200.18	-2.35
LinearGAM	f.s. 3	-2.87	223.56	-2.58
LinearGAM	f.s. 13 ONLY	-3.01	233.36	-2.58
LinearGAM	f.s. 3 ONLY	-3.23	241.21	-2.81
PassiveAggressiveRegressor	f.s. 13 ONLY	-3.65	231.18	-3.07
ExtraTreeRegressor	f.s. 1 ONLY	-4.20	305.10	-3.78
DecisionTreeRegressor	raw feat.	-4.96	323.98	-4.62
ExtraTreeRegressor	raw feat.	-5.28	294.42	-4.85
DecisionTreeRegressor	f.s. 13 ONLY	-5.58	343.45	-5.21
DecisionTreeRegressor	f.s. 2	-5.84	342.08	-5.44
ExtraTreeRegressor	f.s. 13 ONLY	-5.87	403.22	-5.40
PassiveAggressiveRegressor	f.s. 13	-5.97	296.61	-5.35
PassiveAggressiveRegressor	f.s. 1	-5.97	378.34	-4.64
ExtraTreeRegressor	f.s. 13	-6.02	321.34	-5.56
DecisionTreeRegressor	f.s. 1 ONLY	-6.07	333.53	-5.58
LinearGAM	f.s. 1	-6.23	321.23	-5.90
ExtraTreeRegressor	f.s. 1	-6.35	313.28	-5.97
ExtraTreeRegressor	f.s. 2	-6.44	362.21	-6.04
DecisionTreeRegressor	f.s. 3	-6.48	348.34	-6.04
DecisionTreeRegressor	f.s. 1	-6.74	344.55	-6.35
ExtraTreeRegressor	f.s. 3	-6.79	335.34	-6.39
DecisionTreeRegressor	f.s. 13	-7.20	388.09	-6.57
ExtraTreeRegressor	f.s. 3 ONLY	-7.53	335.05	-6.84
LinearGAM	f.s. 13	-7.98	384.95	-7.40
AdaBoostRegressor	f.s. 13	-8.86	341.96	-1.24
PassiveAggressiveRegressor	f.s. 3 ONLY	-9.22	402.45	-8.83
AdaBoostRegressor	f.s. 1	-9.45	343.59	-1.53
DecisionTreeRegressor	f.s. 3 ONLY	-9.51	389.86	-8.47
AdaBoostRegressor	f.s. 13 ONLY	-9.69	357.36	-1.66
PassiveAggressiveRegressor	f.s. 3	-9.93	397.79	-9.59
AdaBoostRegressor	f.s. 3	-10.91	377.75	-0.91
AdaBoostRegressor	f.s. 1 ONLY	-12.02	408.58	-2.39
AdaBoostRegressor	raw feat.	-14.05	453.66	-3.40
AdaBoostRegressor	f.s. 3 ONLY	-17.08	585.41	-0.85

AdaBoostRegressor	f.s. 2	-19.60	600.54	-2.11

3.2 Classifiers

The new way of splitting data set significantly decreases the quality of prediction and it is hard to find a ideal regression model which fits the problem well. So the following work focuses on finding a promising classification model.

3.2.1 Metrics

In order to quantify the quality of predictions, the following metrics are used.

- Balanced Accuracy

  Average of each class's accuracy.

  If tp is the number of true positives, fn the number of false negatives, tn is the number of true negatives and fp the number of false positives, the balanced accuracy is defined as:

$$\texttt{balanced-accuracy} = \frac{1}{2}\left(\frac{tp}{tp+fn} + \frac{tn}{tn+fp}\right) \tag{3.4}$$

- Precision

  The ratio of the number of true positives to the number of all samples that is predicted as positives.

  If tp is the number of true positives and fp the number of false positives, the precision is defined as:

$$\texttt{precision} = \frac{tp}{tp+fp} \tag{3.5}$$

- Sensitivity

  The ratio of the number of true positives to the number of all samples with positive true values.

  If tp is the number of true positives and fn the number of false negatives, the precision is defined as:

$$\texttt{sensitivity} = \frac{tp}{tp+fn} \tag{3.6}$$

- Specificity

  The ratio of the number of true negatives to the number of all samples with negative true values.

  If tn is the number of true negatives and fp the number of false positives, the precision is defined as:

$$\texttt{specificity} = \frac{tn}{tn + fp} \tag{3.7}$$

3.2.2 Labels of the data set

To reduce the original problem to a classification problem labels of data should be created.

An intuitional way to do that is that select a threshold to divide the whole data set into two classes and the samples whose *No. ECB* is less than the threshold are labeled as 0 (negative) and the rest is labeled as 1 (positive).

According to the practical implication of *No. ECB*, the threshold is 10 in this thesis.

Table 3.12 shows the number of positives if the threshold is 10.

Table 3.12: The number of positives

Dataset	#positives	#samples	positives%	negatives%
raw feat.	403	2778	0.1451	0.8549
f.s. 1	403	2665	0.151	0.849
f.s. 1 ONLY	403	2665	0.151	0.849
f.s. 13	403	2551	0.158	0.842
f.s. 13 ONLY	403	2551	0.158	0.842
f.s. 2	403	2665	0.151	0.849
f.s. 3	403	2653	0.152	0.848
f.s. 3 ONLY	403	2653	0.152	0.848

3.2.3 Results of classification methods

All classifiers with the parameters assigned in section 2.3 (*SVC* uses linear kernel and the rest classifiers use their default parameters) are evaluated in the same way as what is done in section 3.1 and the results are shown in the Table 3.13 and Table 3.14.

To limit the width of the tables, the following abbreviations are used:

Accu. = Balanced Accuracy, Pre. = Precision, Sen. = Sensitivity, Spe. = Specificity

Table 3.13: Results of classifiers with 10-fold

Model	Dataset	Accu.	Pre.	Sen.	Spe.
ExtraTreesClassifier	f.s. 13 ONLY	0.74	0.74	0.53	0.94
RandomForestClassifier	f.s. 13 ONLY	0.73	0.75	0.51	0.94
GradientBoostingClassifier	f.s. 13 ONLY	0.72	0.69	0.52	0.92
GradientBoostingClassifier	f.s. 13	0.72	0.68	0.51	0.92
KNeighborsClassifier	f.s. 13 ONLY	0.71	0.60	0.54	0.88
RandomForestClassifier	f.s. 13	0.70	0.72	0.45	0.94
BaggingClassifier	f.s. 13 ONLY	0.70	0.67	0.49	0.92
AdaBoostClassifier	f.s. 13 ONLY	0.70	0.61	0.50	0.89
BaggingClassifier	f.s. 13	0.69	0.64	0.46	0.91
AdaBoostClassifier	f.s. 13	0.69	0.58	0.49	0.89
DecisionTreeClassifier	f.s. 13 ONLY	0.69	0.52	0.55	0.84
DecisionTreeClassifier	f.s. 13	0.69	0.52	0.54	0.84
ExtraTreesClassifier	f.s. 13	0.68	0.70	0.42	0.94
KNeighborsClassifier	f.s. 13	0.68	0.57	0.48	0.88
GaussianNB	f.s. 13 ONLY	0.68	0.37	0.83	0.53
GaussianNB	f.s. 1 ONLY	0.68	0.35	0.86	0.50
ExtraTreesClassifier	f.s. 1 ONLY	0.67	0.69	0.40	0.94
RandomForestClassifier	f.s. 1 ONLY	0.67	0.68	0.40	0.94
GradientBoostingClassifier	f.s. 1 ONLY	0.67	0.66	0.40	0.94
AdaBoostClassifier	f.s. 1	0.67	0.59	0.42	0.91
KNeighborsClassifier	f.s. 1 ONLY	0.67	0.55	0.47	0.88
GaussianNB	f.s. 13	0.67	0.35	0.84	0.49
GradientBoostingClassifier	f.s. 1	0.66	0.65	0.40	0.93
BaggingClassifier	f.s. 1 ONLY	0.66	0.64	0.39	0.93
AdaBoostClassifier	f.s. 1 ONLY	0.66	0.58	0.41	0.91
ExtraTreeClassifier	f.s. 13 ONLY	0.66	0.48	0.50	0.82
GaussianNB	raw feat.	0.66	0.36	0.84	0.48
GaussianNB	f.s. 1	0.66	0.34	0.86	0.47
ExtraTreesClassifier	f.s. 1	0.65	0.67	0.35	0.95
DecisionTreeClassifier	f.s. 1 ONLY	0.65	0.45	0.49	0.81
RandomForestClassifier	f.s. 1	0.64	0.63	0.35	0.94
BaggingClassifier	f.s. 1	0.64	0.61	0.34	0.93
SVC	f.s. 13	0.64	0.56	0.39	0.90
KNeighborsClassifier	f.s. 1	0.64	0.51	0.40	0.88

DecisionTreeClassifier	f.s. 1	0.64	0.44	0.47	0.81
ExtraTreeClassifier	f.s. 1 ONLY	0.63	0.43	0.44	0.81
AdaBoostClassifier	f.s. 3	0.62	0.51	0.35	0.89
SVC	f.s. 3	0.62	0.48	0.35	0.88
ExtraTreeClassifier	f.s. 13	0.62	0.42	0.44	0.80
SVC	f.s. 13 ONLY	0.61	0.58	0.28	0.93
GradientBoostingClassifier	raw feat.	0.60	0.60	0.25	0.95
GradientBoostingClassifier	f.s. 3	0.60	0.59	0.26	0.94
RandomForestClassifier	raw feat.	0.60	0.57	0.26	0.93
AdaBoostClassifier	raw feat.	0.60	0.52	0.28	0.91
KNeighborsClassifier	raw feat.	0.60	0.47	0.34	0.87
DecisionTreeClassifier	raw feat.	0.60	0.40	0.41	0.79
ExtraTreeClassifier	f.s. 1	0.60	0.38	0.41	0.79
KNeighborsClassifier	f.s. 3	0.60	0.38	0.41	0.78
ExtraTreesClassifier	raw feat.	0.59	0.57	0.25	0.93
BaggingClassifier	raw feat.	0.59	0.50	0.26	0.91
ExtraTreeClassifier	f.s. 3	0.59	0.37	0.41	0.78
SVC	f.s. 1	0.58	0.49	0.21	0.95
ExtraTreeClassifier	raw feat.	0.58	0.37	0.37	0.78
DecisionTreeClassifier	f.s. 3	0.58	0.36	0.37	0.79
GaussianNB	f.s. 3	0.58	0.29	0.80	0.37
ExtraTreesClassifier	f.s. 3	0.56	0.62	0.15	0.97
RandomForestClassifier	f.s. 3	0.56	0.59	0.15	0.97
BaggingClassifier	f.s. 3	0.56	0.46	0.20	0.92
DecisionTreeClassifier	f.s. 2	0.56	0.33	0.37	0.76
GaussianNB	f.s. 3 ONLY	0.56	0.28	0.78	0.35
GaussianNB	f.s. 2	0.56	0.27	0.82	0.29
RandomForestClassifier	f.s. 2	0.55	0.52	0.14	0.96
KNeighborsClassifier	f.s. 2	0.55	0.36	0.23	0.87
ExtraTreeClassifier	f.s. 2	0.55	0.31	0.34	0.76
ExtraTreesClassifier	f.s. 2	0.54	0.51	0.12	0.96
BaggingClassifier	f.s. 2	0.54	0.41	0.16	0.93
GradientBoostingClassifier	f.s. 2	0.53	0.47	0.08	0.97
AdaBoostClassifier	f.s. 2	0.53	0.42	0.10	0.96
KNeighborsClassifier	f.s. 3 ONLY	0.53	0.29	0.32	0.75
AdaBoostClassifier	f.s. 3 ONLY	0.52	0.32	0.11	0.92
RandomForestClassifier	f.s. 3 ONLY	0.51	0.42	0.02	1.00
ExtraTreesClassifier	f.s. 3 ONLY	0.51	0.38	0.02	0.99
BaggingClassifier	f.s. 3 ONLY	0.51	0.30	0.07	0.95

DecisionTreeClassifier	f.s. 3 ONLY	0.51	0.27	0.29	0.74
ExtraTreeClassifier	f.s. 3 ONLY	0.50	0.25	0.27	0.73
GradientBoostingClassifier	f.s. 3 ONLY	0.50	0.13	0.01	0.99
SVC	raw feat.	0.50	0.00	0.00	1.00
SVC	f.s. 1 ONLY	0.50	0.00	0.00	1.00
SVC	f.s. 2	0.50	0.00	0.00	1.00
SVC	f.s. 3 ONLY	0.50	0.00	0.00	1.00

Table 3.14: Results of classifiers with leave one year out

Model	Dataset	Accu.	Pre.	Sen.	Spe.
GaussianNB	f.s. 13 ONLY	0.68	0.38	0.84	0.51
GaussianNB	f.s. 1 ONLY	0.68	0.37	0.85	0.51
AdaBoostClassifier	f.s. 13	0.66	0.54	0.44	0.87
GaussianNB	f.s. 13	0.66	0.37	0.85	0.48
GaussianNB	f.s. 1	0.66	0.36	0.85	0.48
GaussianNB	raw feat.	0.66	0.35	0.83	0.50
SVC	f.s. 13	0.63	0.53	0.36	0.89
AdaBoostClassifier	f.s. 13 ONLY	0.63	0.49	0.40	0.86
GradientBoostingClassifier	f.s. 13	0.62	0.54	0.34	0.91
AdaBoostClassifier	f.s. 1 ONLY	0.62	0.50	0.36	0.88
AdaBoostClassifier	f.s. 1	0.62	0.49	0.36	0.88
GradientBoostingClassifier	f.s. 1	0.61	0.54	0.31	0.91
GradientBoostingClassifier	f.s. 13 ONLY	0.61	0.52	0.32	0.90
BaggingClassifier	f.s. 13 ONLY	0.61	0.52	0.30	0.91
SVC	f.s. 3	0.61	0.46	0.33	0.89
KNeighborsClassifier	f.s. 13	0.61	0.45	0.38	0.85
KNeighborsClassifier	f.s. 13 ONLY	0.61	0.42	0.39	0.82
DecisionTreeClassifier	f.s. 13 ONLY	0.61	0.39	0.42	0.79
DecisionTreeClassifier	f.s. 1	0.61	0.38	0.45	0.76
BaggingClassifier	f.s. 13	0.60	0.52	0.30	0.90
GradientBoostingClassifier	f.s. 1 ONLY	0.60	0.51	0.29	0.91
ExtraTreesClassifier	f.s. 13 ONLY	0.60	0.51	0.27	0.92
RandomForestClassifier	f.s. 1	0.60	0.50	0.27	0.93
AdaBoostClassifier	f.s. 3	0.60	0.49	0.31	0.89
RandomForestClassifier	f.s. 13	0.60	0.47	0.29	0.91
RandomForestClassifier	f.s. 13 ONLY	0.60	0.47	0.29	0.91
KNeighborsClassifier	f.s. 1 ONLY	0.60	0.45	0.35	0.85

KNeighborsClassifier	f.s. 1	0.60	0.44	0.34	0.86
DecisionTreeClassifier	f.s. 13	0.60	0.39	0.42	0.78
AdaBoostClassifier	raw feat.	0.59	0.51	0.26	0.92
RandomForestClassifier	f.s. 1 ONLY	0.59	0.51	0.25	0.93
ExtraTreesClassifier	f.s. 13	0.59	0.50	0.24	0.94
ExtraTreesClassifier	raw feat.	0.59	0.50	0.24	0.93
BaggingClassifier	raw feat.	0.59	0.49	0.26	0.91
ExtraTreeClassifier	raw feat.	0.59	0.37	0.39	0.79
GaussianNB	f.s. 3	0.59	0.31	0.80	0.38
GradientBoostingClassifier	raw feat.	0.58	0.57	0.22	0.94
RandomForestClassifier	raw feat.	0.58	0.54	0.22	0.94
GradientBoostingClassifier	f.s. 3	0.58	0.52	0.22	0.94
ExtraTreesClassifier	f.s. 1	0.58	0.49	0.23	0.93
ExtraTreesClassifier	f.s. 1 ONLY	0.58	0.45	0.22	0.94
KNeighborsClassifier	raw feat.	0.58	0.41	0.28	0.87
BaggingClassifier	f.s. 1 ONLY	0.58	0.41	0.25	0.90
ExtraTreeClassifier	f.s. 13 ONLY	0.58	0.38	0.39	0.77
ExtraTreeClassifier	f.s. 1 ONLY	0.58	0.37	0.37	0.79
ExtraTreeClassifier	f.s. 13	0.58	0.36	0.41	0.75
DecisionTreeClassifier	raw feat.	0.58	0.35	0.37	0.78
BaggingClassifier	f.s. 1	0.57	0.42	0.24	0.90
SVC	f.s. 13 ONLY	0.57	0.41	0.18	0.95
KNeighborsClassifier	f.s. 3	0.57	0.36	0.36	0.78
DecisionTreeClassifier	f.s. 3	0.57	0.36	0.35	0.78
DecisionTreeClassifier	f.s. 1 ONLY	0.57	0.35	0.36	0.78
ExtraTreeClassifier	f.s. 1	0.57	0.35	0.36	0.78
ExtraTreeClassifier	f.s. 3	0.57	0.34	0.37	0.77
GaussianNB	f.s. 3 ONLY	0.57	0.29	0.77	0.36
RandomForestClassifier	f.s. 3	0.56	0.49	0.16	0.96
SVC	f.s. 1	0.56	0.43	0.18	0.95
ExtraTreesClassifier	f.s. 3	0.55	0.56	0.14	0.97
BaggingClassifier	f.s. 3	0.55	0.41	0.18	0.92
GaussianNB	f.s. 2	0.55	0.28	0.80	0.29
DecisionTreeClassifier	f.s. 2	0.54	0.30	0.33	0.75
AdaBoostClassifier	f.s. 2	0.53	0.48	0.10	0.96
ExtraTreesClassifier	f.s. 2	0.53	0.39	0.10	0.96
ExtraTreeClassifier	f.s. 2	0.53	0.30	0.32	0.75
GradientBoostingClassifier	f.s. 2	0.52	0.46	0.06	0.97
BaggingClassifier	f.s. 2	0.52	0.34	0.11	0.94

RandomForestClassifier	f.s. 2	0.52	0.34	0.10	0.95
AdaBoostClassifier	f.s. 3 ONLY	0.52	0.31	0.10	0.93
KNeighborsClassifier	f.s. 2	0.52	0.30	0.18	0.86
DecisionTreeClassifier	f.s. 3 ONLY	0.52	0.29	0.30	0.74
BaggingClassifier	f.s. 3 ONLY	0.51	0.29	0.07	0.95
KNeighborsClassifier	f.s. 3 ONLY	0.51	0.27	0.29	0.73
ExtraTreeClassifier	f.s. 3 ONLY	0.51	0.26	0.27	0.74
RandomForestClassifier	f.s. 3 ONLY	0.50	0.20	0.01	1.00
ExtraTreesClassifier	f.s. 3 ONLY	0.50	0.11	0.00	1.00
GradientBoostingClassifier	f.s. 3 ONLY	0.50	0.08	0.00	0.99
SVC	raw feat.	0.50	0.00	0.00	1.00
SVC	f.s. 1 ONLY	0.50	0.00	0.00	1.00
SVC	f.s. 2	0.50	0.00	0.00	1.00
SVC	f.s. 3 ONLY	0.50	0.00	0.00	1.00

10-fold cross-validation makes the problem easier, which is the same as the situation of regression. When 10-fold is used, the balanced accuracy of 12 combinations of model and data set is greater than 0.68 which is the highest balanced accuracy when "leave one year out" is used.

Besides, adding historical weather information (f.s. 1 and f.s. 13) improves the performance, which is also the same as the situation of regression.

When 10-fold is used, *ExtraTreesClassifier* and *RandomForestClassifier* with *f.s. 13 ONLY* have the best performances but the sensitivities of them are a little more than 0.5. So only about half of the positives are correctly predicted by the models.

When "leave one year out" is used, as just mentioned, the performances of models are worse. *GaussianNB* with *f.s. 1 ONLY*, *f.s 13 ONLY* has the the highest balanced accuracy and very similar performance, however, *AdaBoostClassifier* with *f.s. 13* has a slightly lower balanced accuracy, higher precision, specificity and lower sensitivity. The former models give more positive predictions at the cost of lower precision of positive prediction and accuracy of predicting negatives while the latter one is the opposite.

The predictions given by *SVC* are extreme manifestations of this situation. As Figure 3.1 shows, *SVC* just always gives negative predictions and doesn't learn anything from the training set.

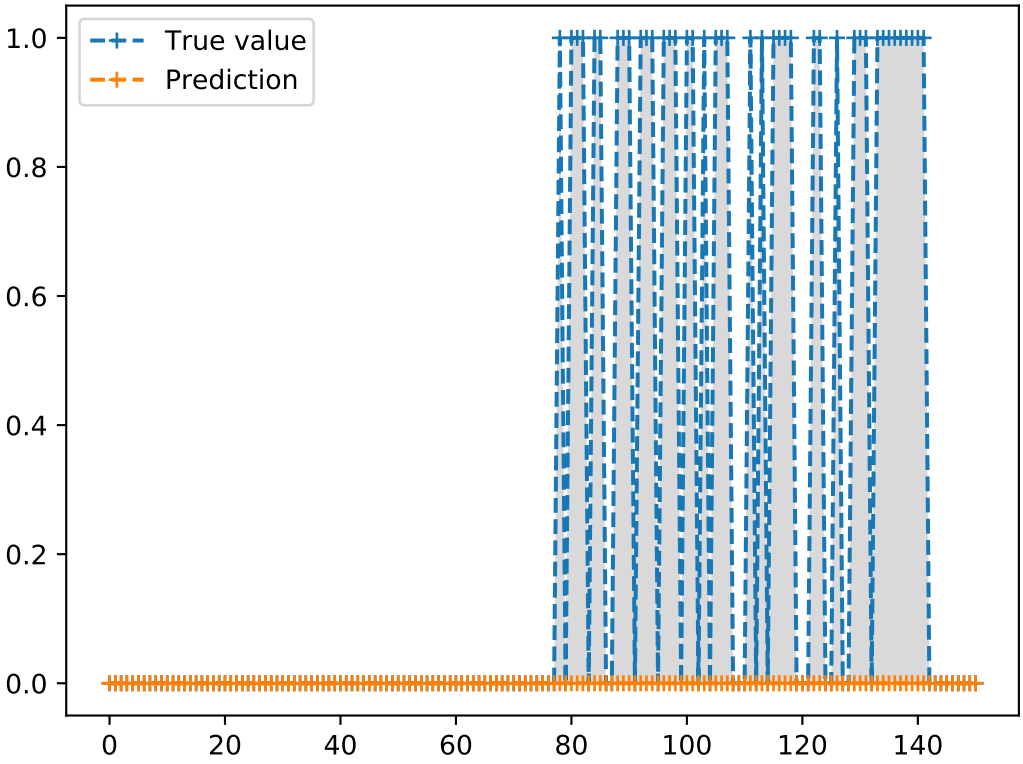


Figure 3.1. Prediction of 1999 by *SVC* with the raw data set and leave one year out

Chapter 4

Conclusion and Future Work

4.1 Conclusion

All algorithms evaluated in this thesis can not be used by the hop farmers, and there are still lots of jobs to be done.

The first attempt, adding historical weather information (especially adding daily mean, daily maximum, daily minimum, daily variance, daily median, mean of the day and the past several days, maximum of the day and the past several days, minimum of the day and the past several days) improves the evaluated performances to some extent whether it's regression problem or classification problem.

The second attempt, reducing the original problem to a classification problem, also works, given that the balanced accuracy of all classifiers is greater than or equal to 0.5 and R^2 scores of all 9 regressors whose classification version algorithm are evaluated in this thesis is less than 0 if 10-fold and raw data set are used and this situation is almost the same if "leave one year out" is used.

Besides, 10-fold shuffles all the data and makes the prediction easier because of the significant performance decrease.

4.2 Future Work

Based on the evaluations that have been done, collecting more data could be helpful. Because there are only 403 positive samples if the threshold is

not changed. And the small data set limits the choice of algorithm especially the neural network algorithms.

Trying other kinds of learning algorithms is also a choice. Considering a appropriate way to add historical weather information improves the performance, those algorithms which can deal with entire sequences of data, such as LSTM, may have advantages. But given that many of that kind of algorithms are based on neural network so it is still need to be studied.

Combining two different kinds of models may also be helpful.

Finally, adding the number of captured ECB in the last few days to the training process of models may be a good attempt. In the reality, not only weather condition but also the historical population number affects the current number of ECB.

Appendix A.

Figures of the data set

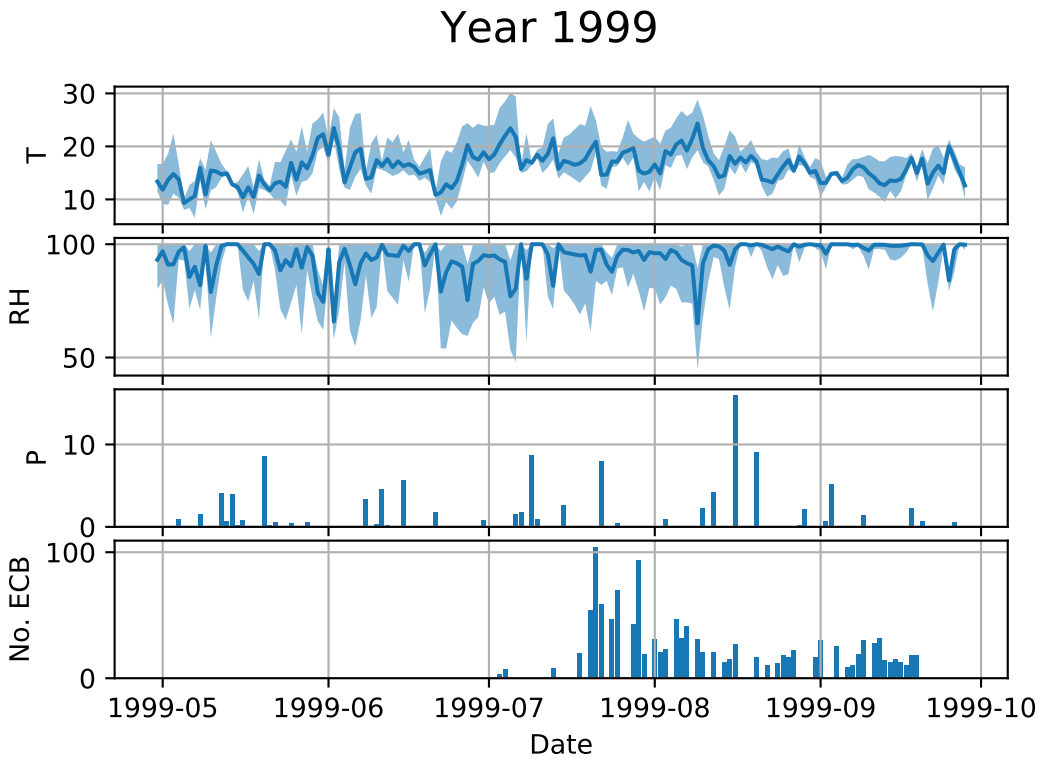
The following are figures of each year's data.

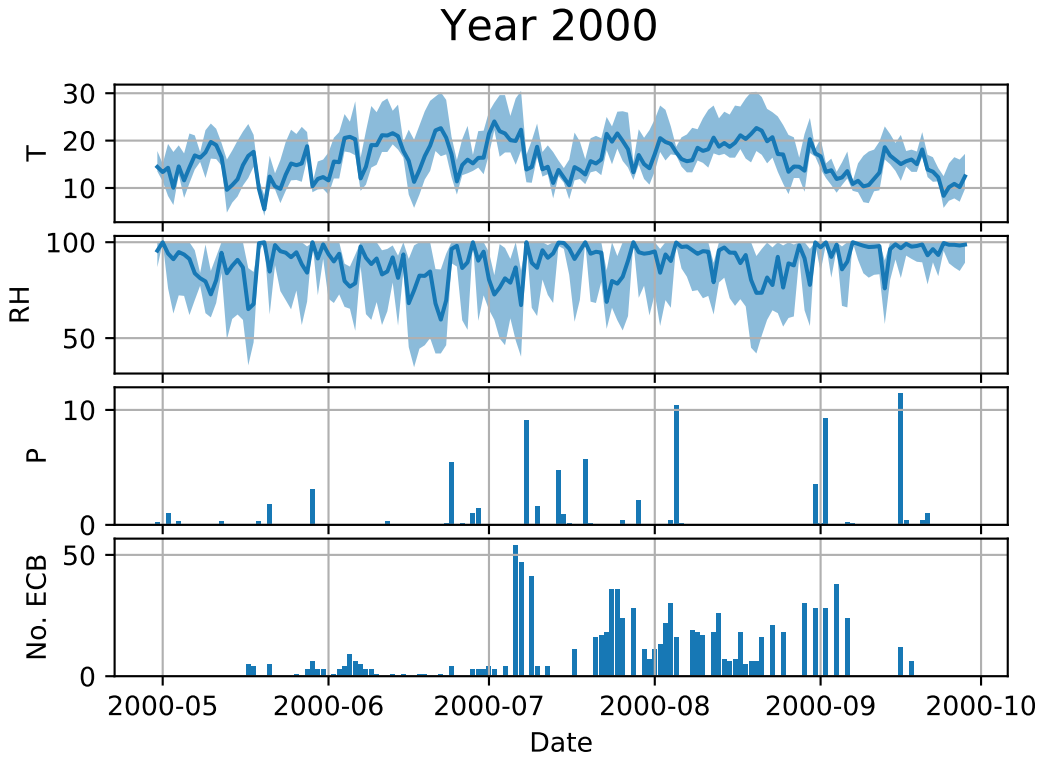
T is temperature.

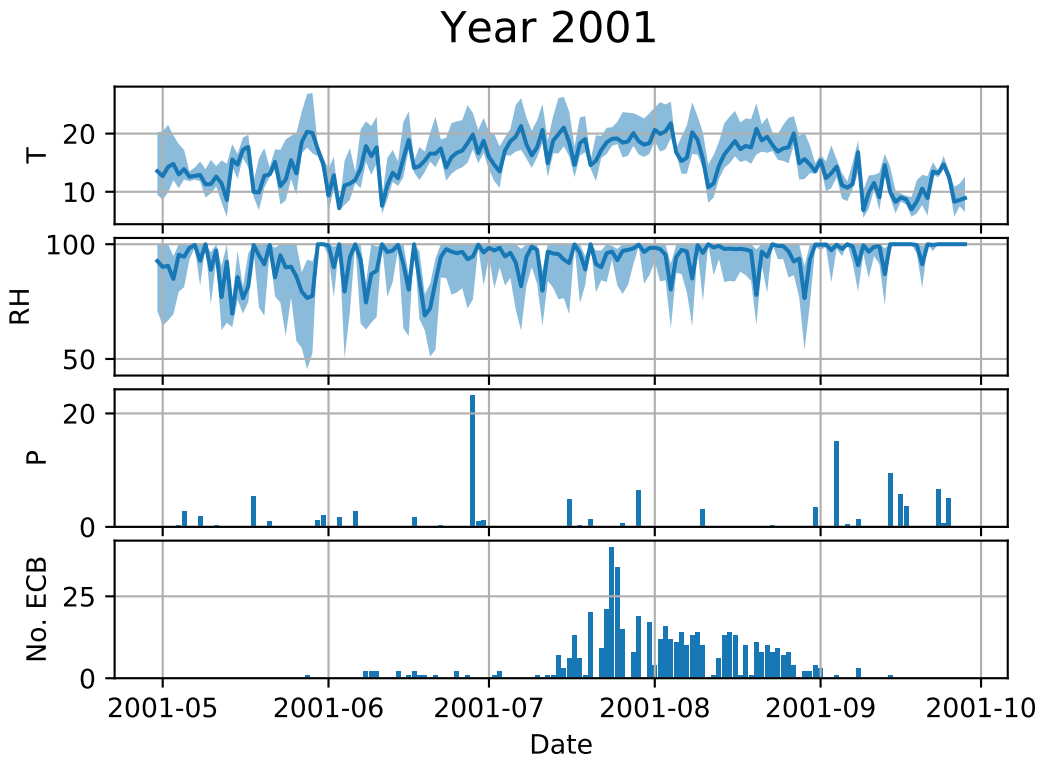
RH is relative humidity. The blue line is the mean of each day and the blue shadow is filled between the maximum and the minimum of each day.

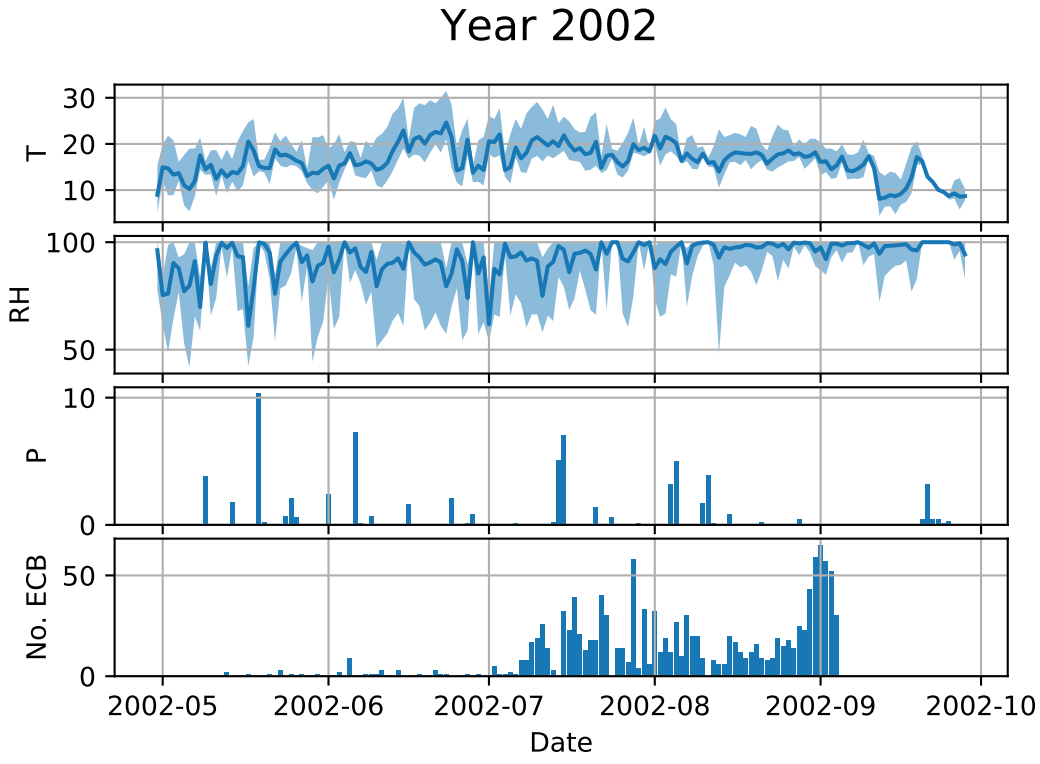
P is the total precipitation of each day.

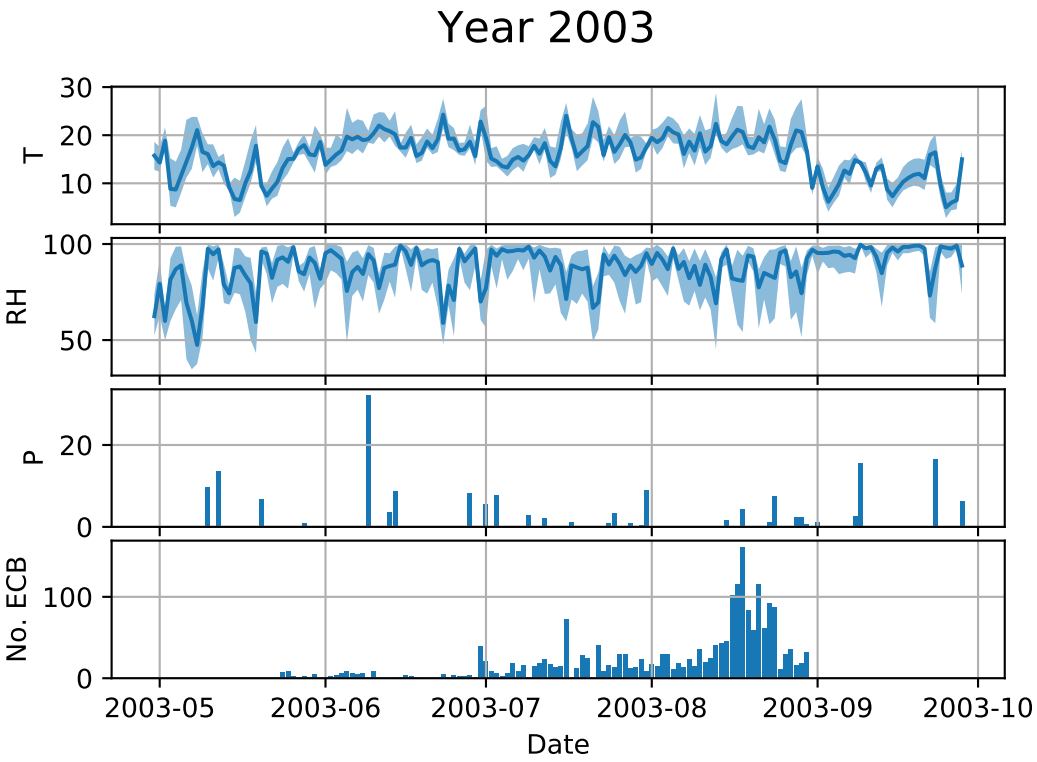
No.ECB is the number of ECB captured of each day.

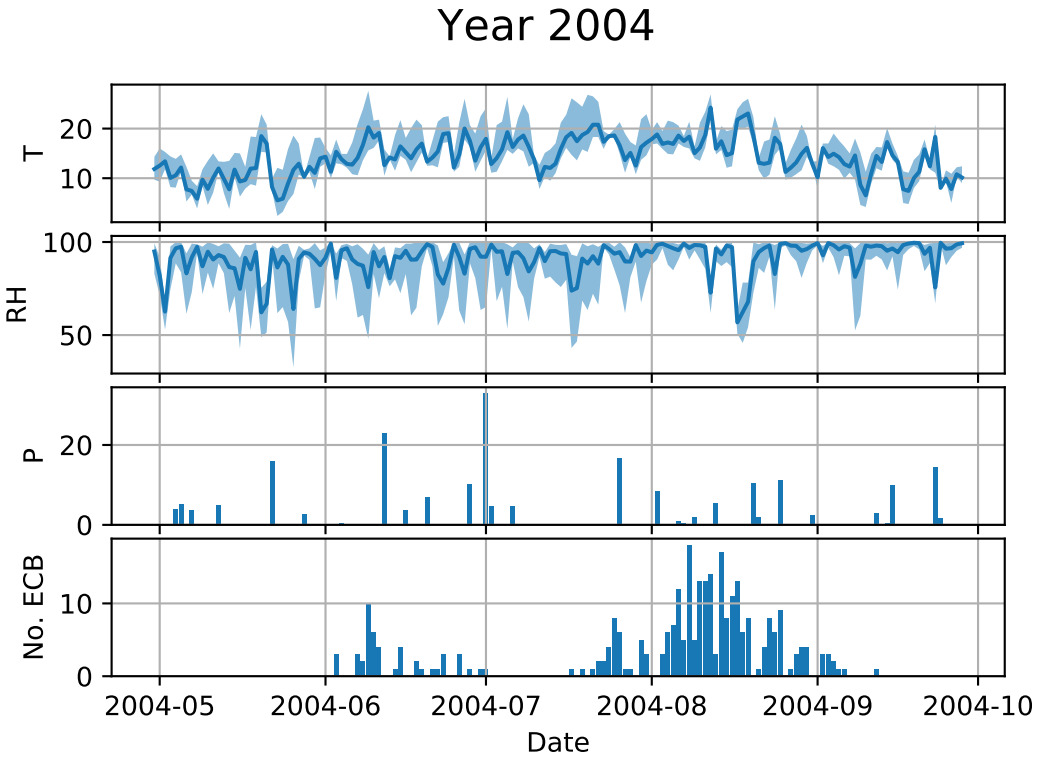

Year 1999
T
RH
P
No. ECB
30
20
10
100
50
10
0
100
0
1999-05
1999-06
1999-07
1999-08
1999-09
1999-10
Date

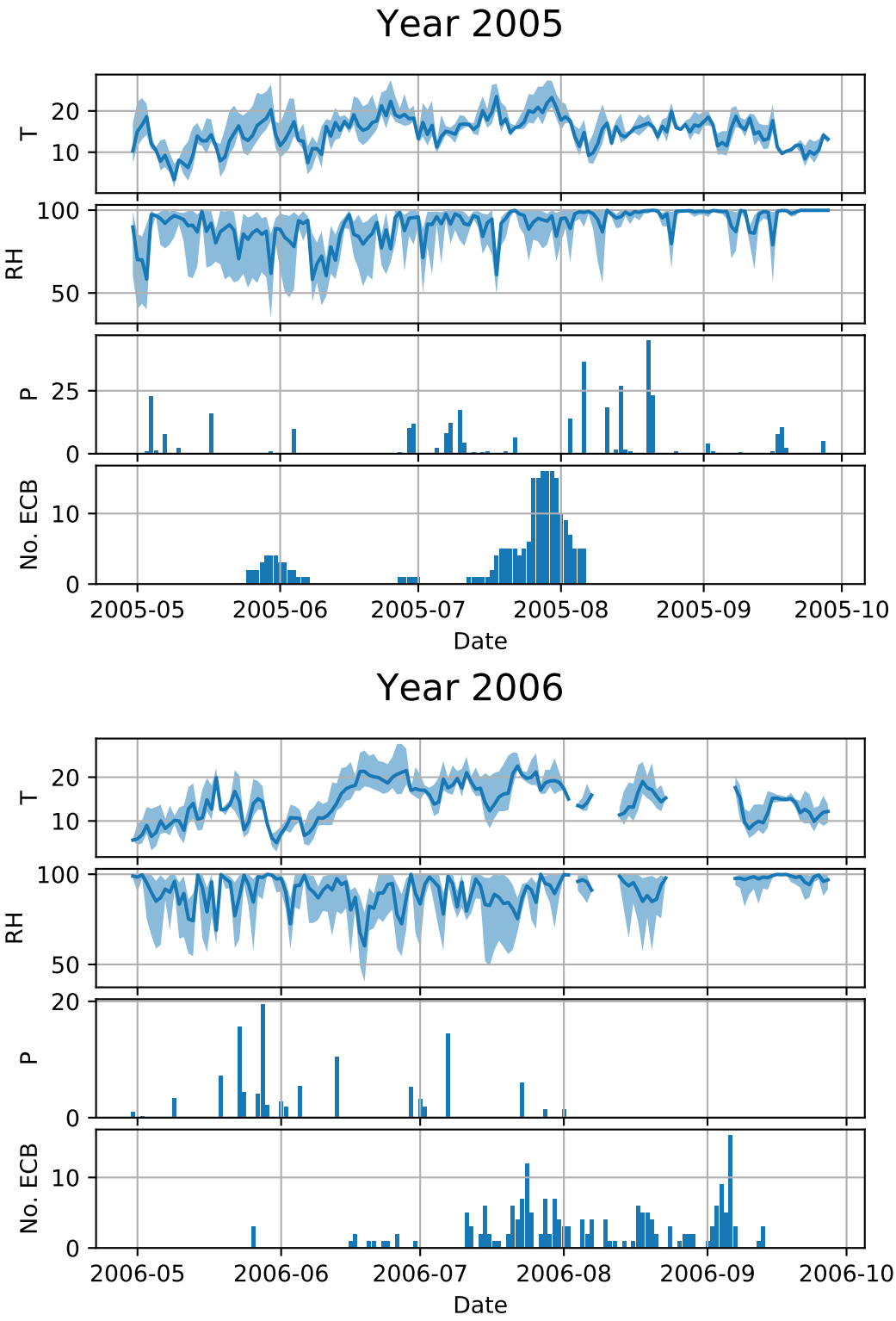

Year 2000
T
RH
P
No. ECB
30
20
10
100
50
10
0
50
0
2000-05
2000-06
2000-07
2000-08
2000-09
2000-10
Date

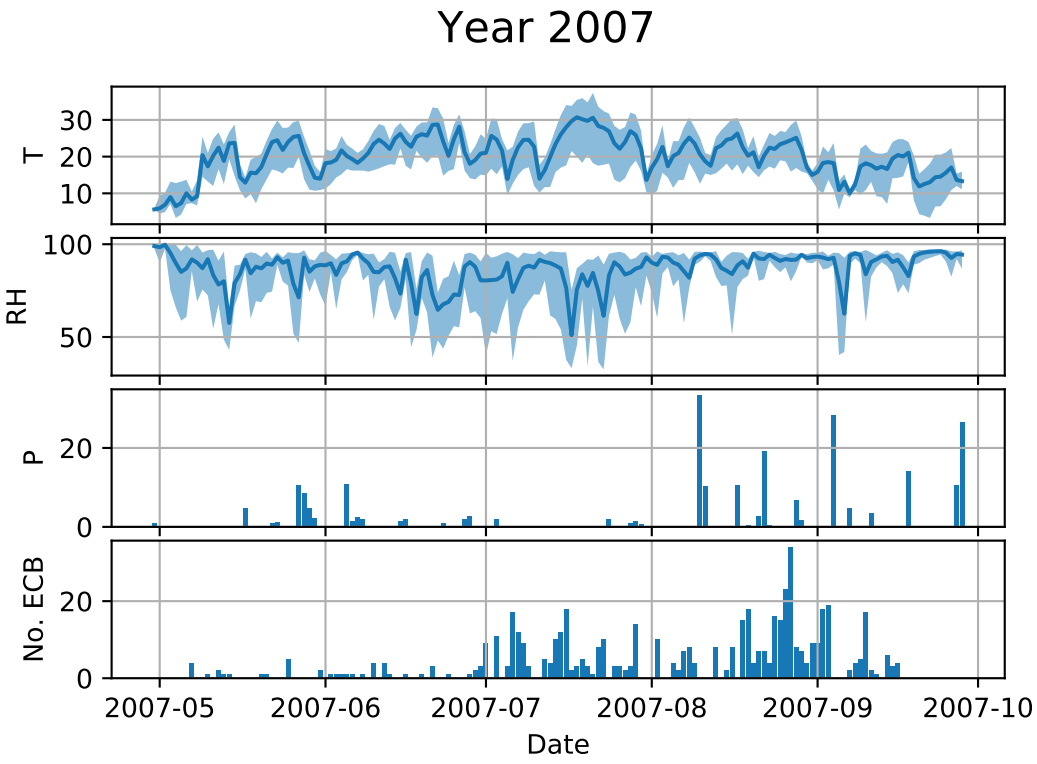

Year 2001
T
RH
P
No. ECB
2001-05
2001-06
2001-07
2001-08
2001-09
2001-10
Date

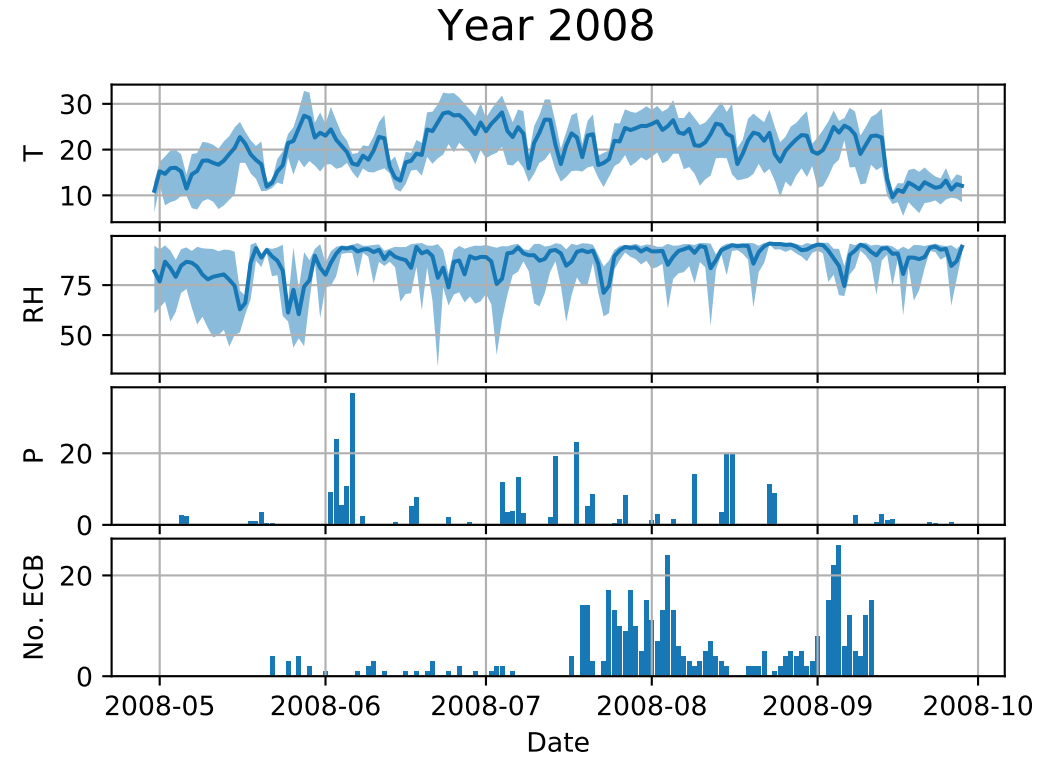

Year 2002
T
RH
P
No. ECB
2002-05
2002-06
2002-07
2002-08
2002-09
2002-10
Date

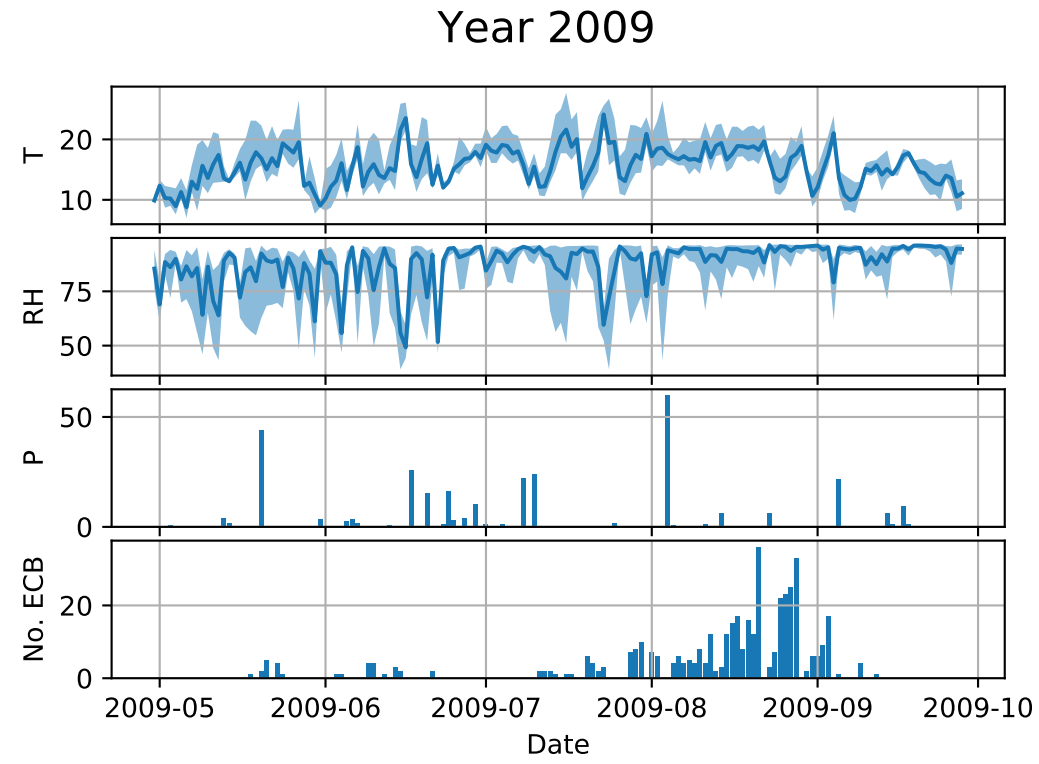

Year 2003
T
RH
P
No. ECB
2003-05
2003-06
2003-07
2003-08
2003-09
2003-10
Date

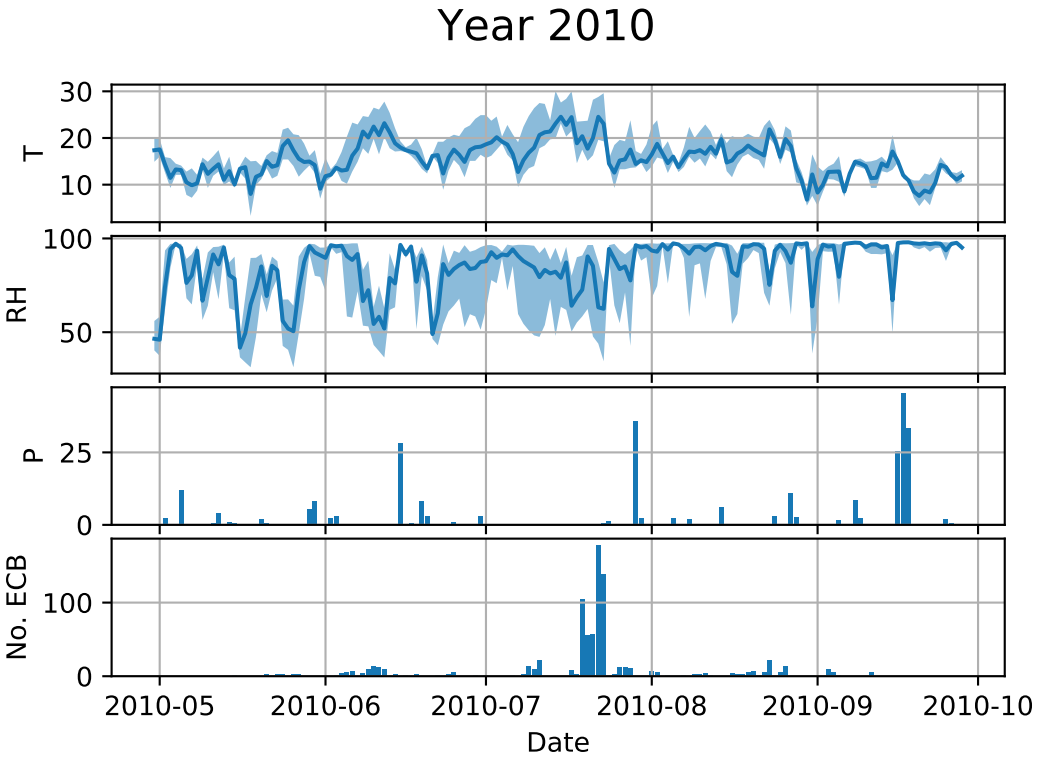

Year 2004
T
RH
P
No. ECB
2004-05
2004-06
2004-07
2004-08
2004-09
2004-10
Date

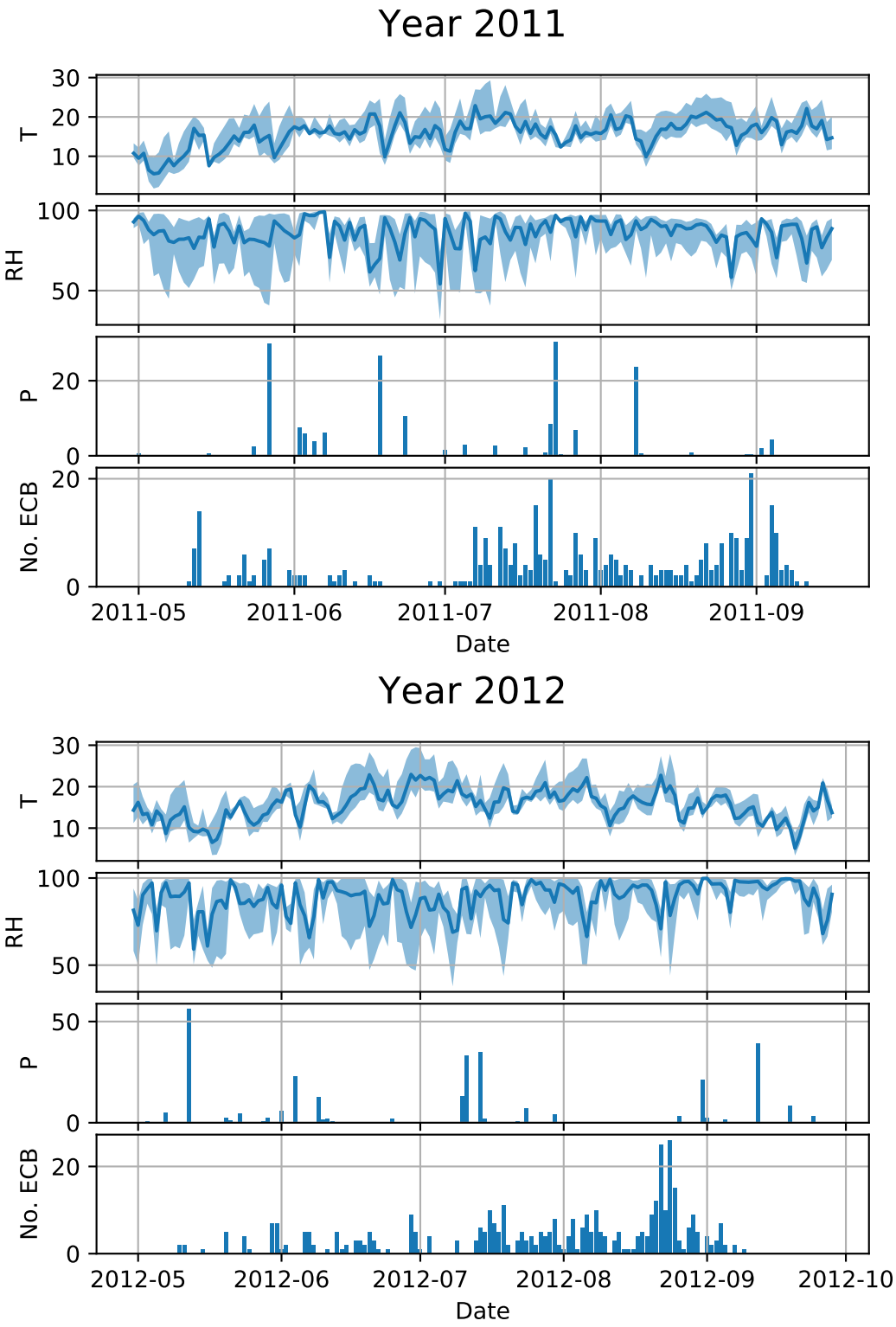

Year 2005
T
RH
P
No. ECB
2005-05
2005-06
2005-07
2005-08
2005-09
2005-10
Date
Year 2006
T
RH
P
No. ECB
2006-05
2006-06
2006-07
2006-08
2006-09
2006-10
Date

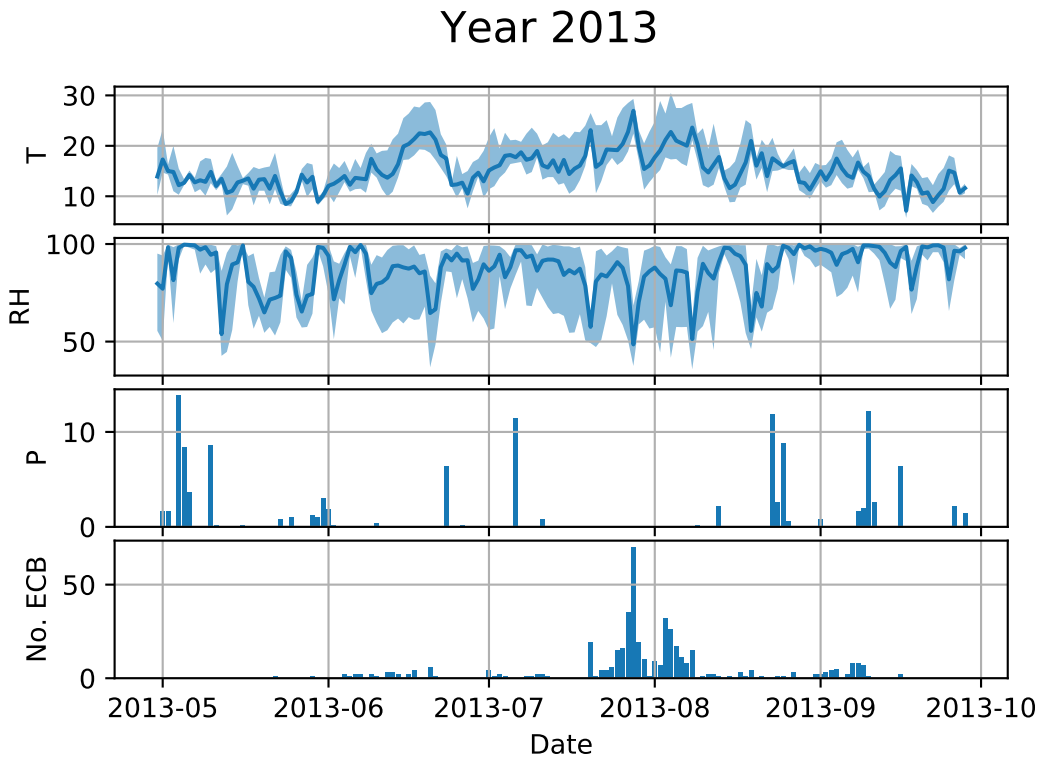

Year 2007
T
30
20
10
RH
100
50
P
20
0
No. ECB
20
0
2007-05
2007-06
2007-07
2007-08
2007-09
2007-10
Date

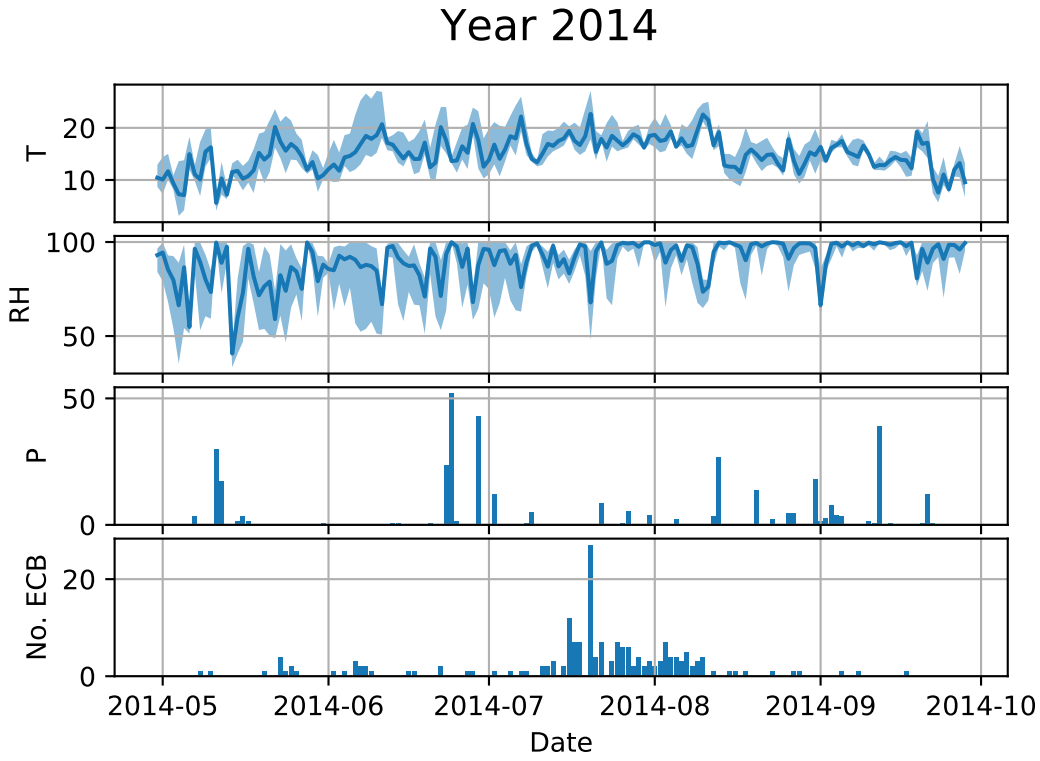

Year 2008
T
30
20
10
RH
75
50
P
20
0
No. ECB
20
0
2008-05
2008-06
2008-07
2008-08
2008-09
2008-10
Date

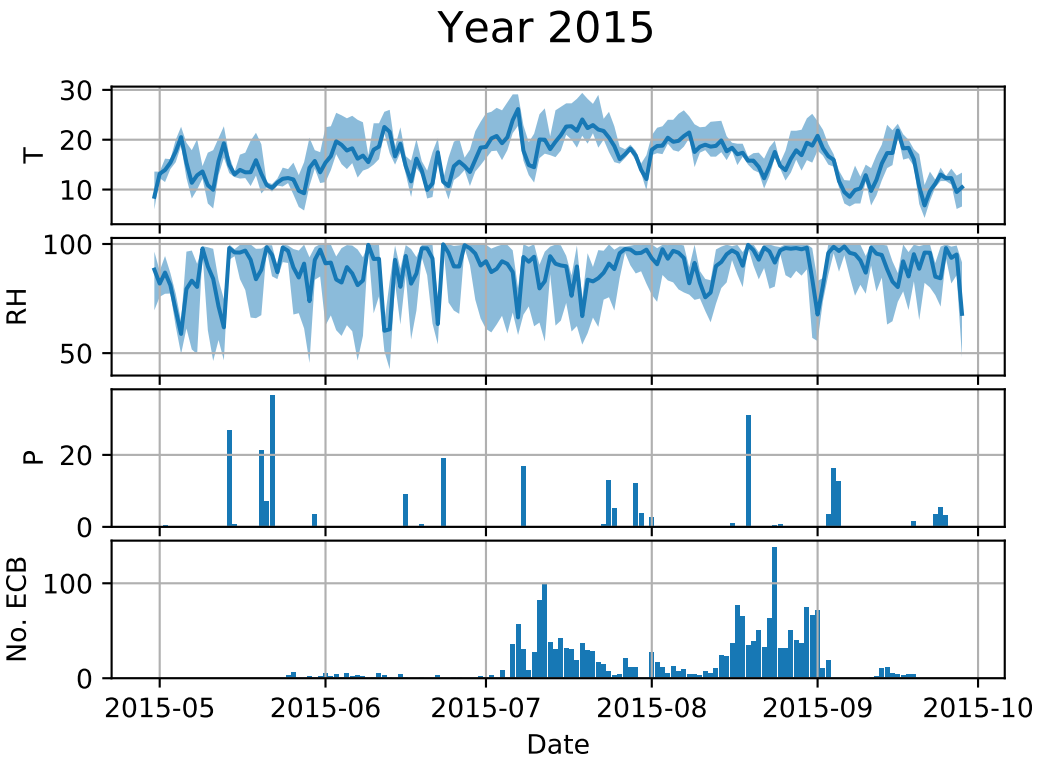

Year 2009
T
20
10
RH
75
50
P
50
0
No. ECB
20
0
2009-05
2009-06
2009-07
2009-08
2009-09
2009-10
Date

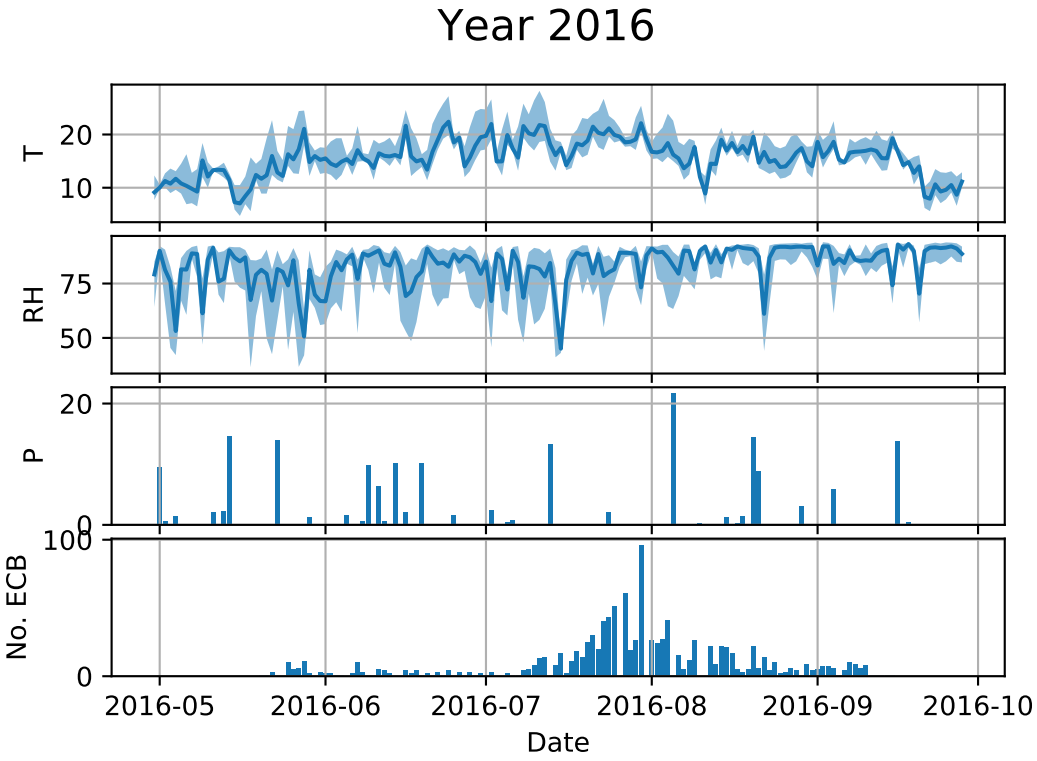

Year 2010
T
30
20
10
RH
100
50
P
25
0
No. ECB
100
0
2010-05
2010-06
2010-07
2010-08
2010-09
2010-10
Date

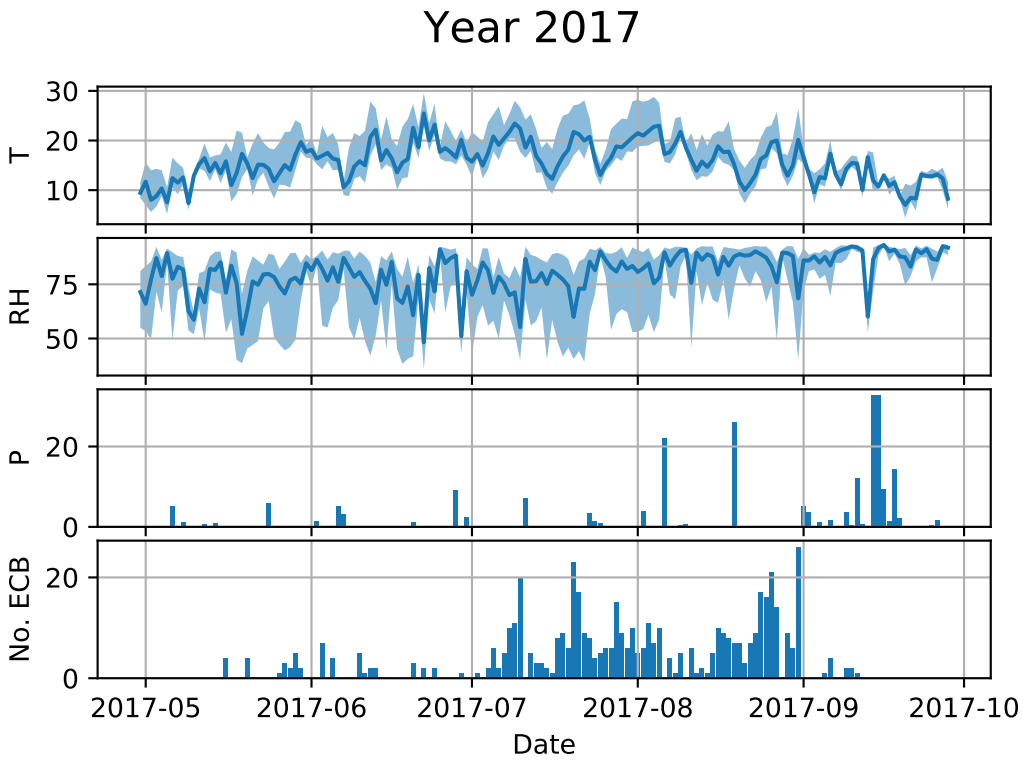

Year 2011
T
RH
P
No. ECB
Date
2011-05
2011-06
2011-07
2011-08
2011-09
Year 2012
2012-05
2012-06
2012-07
2012-08
2012-09
2012-10


Year 2013
T
RH
P
No. ECB
2013-05
2013-06
2013-07
2013-08
2013-09
2013-10
Date


Year 2014
T
RH
P
No. ECB
2014-05
2014-06
2014-07
2014-08
2014-09
2014-10
Date


Year 2015
T
RH
P
No. ECB
Date
2015-05
2015-06
2015-07
2015-08
2015-09
2015-10


Year 2016
T
RH
P
No. ECB
Date
2016-05
2016-06
2016-07
2016-08
2016-09
2016-10


Year 2017
T
RH
P
No. ECB
30
20
10
75
50
20
0
20
0
2017-05
2017-06
2017-07
2017-08
2017-09
2017-10
Date

Bibliography

[1] MR Cizej, Pasquale Trematerra, et al. Flight patterns of the european corn borer, ostrinia nubilalis, in slovenian hop gardens in 1999-2016. *Bulletin of Insectology*, 70(2):299–305, 2017.

[2] C JANUS and M STENGEL. La pyrale du mais sur houblon observations et interrogations. *Phytoma*, 25:27–30, 1973.

[3] Donald John Caffrey and Leon Howard Worthley. *A progress report on the investigations of the European corn borer*. Number 1476. US Government Printing Office, 1927.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Daniel Servén and Charlie Brummitt. pygam: Generalized additive models in python, March 2018.

[6] Jacob A Wegelin et al. A survey of partial least squares (pls) methods, with emphasis on the two-block case. *University of Washington, Tech. Rep*, 2000.

[7] Svante Wold, Arnold Ruhe, Herman Wold, and WJ Dunn, III. The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

[8] Matthias Otto and Wolfhard Wegscheider. Spectrophotometric multicomponent analysis applied to trace metal determinations. *Analytical Chemistry*, 57(1):63–69, 1985.

[9] Harris Drucker. Improving regressors using boosting techniques. In *ICML*, volume 97, pages 107–115, 1997.

[10] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization

of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[11] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine learning*, 36(1-2):85–103, 1999.

[12] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[13] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[14] Gilles Louppe and Pierre Geurts. Ensembles on random patches. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 346–361. Springer, 2012.

[15] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[16] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[17] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[18] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[19] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[20] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[21] David P Wipf and Srikantan S Nagarajan. A new view of automatic relevance determination. In *Advances in neural information processing systems*, pages 1625–1632, 2008.

[22] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[23] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[24] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[25] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[26] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[27] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[28] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[29] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[30] Trevor J Hastie and Robert J Tibshirani. *Generalized additive models*, volume 43. CRC press, 1990.

[31] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[32] Robert GD Steel and James H Torrie. Principles and procedures of statistcs with special reference to the biological sciences. Technical report, McGraw-Hill Book Company, Inc., 1960.

[33] Stanton A Glantz, Bryan K Slinker, and Torsten B Neilands. *Primer of applied regression and analysis of variance*, volume 309. McGraw-Hill New York, 1990.

[34] Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

[35] Gloria Rosenthal and James A Rosenthal. *Statistics and data interpretation for social work*. Springer publishing company, 2011.