Exercise 1 - Classification
Machine Learning

Autors: Jan Bolcek, Isil Isiksalan
Date: 11.11.2021

1 Task Description

The goal of this exercise is to create and learn Machine Learning models to be able to solve classification tasks.

2 Description of used classifiers

In these section a brief description of used classifiers will be made. For all of models implementations the python open-source scikit-learn library is used.

2.1 Gaussian Naive Bayes

Gaussian Naive Bayes is one of the most simple algorithm for performing classification. It works on the Bayes theorem. One of the big advantage is that there are no hyper-parameters to tune and it is very easy to implement. On the other hand it would not be sufficient for solving complex problems.

2.2 Random Forest

Random forest could be used, for classification and also for regression tasks. It consist of n decision trees, that solve the task individually. The final result is the class selected by most tree. There are several hyperparameters that could control the performance of Random forest classifier, in this exercise the following parameters would be tuned:

- **n_estimators** - number of trees in the forest
- **max_features** - number of trees in the forest
- **max depth** - The maximum depth of the tree
- **criterion** - The function to measure the quality of a split

2.3 Support Vector Machine

The task of the Support Vector Machine (SVC) is to create a hyper-plane in N-dimensional space, that would be able to distinctly classifies the data points.There are several hyperparameters that could control the performance of SVC, in this exercise the following parameters would be tuned:

- **C** - Regularization parameter
- **Gamma** - coeficinents for kernels
- **Kernel** - Specifies the kernel type to be used in the algorithm

3 Descriptions of used metrics

In Machine learning, the performance of trained model needs to be validate. For that purpose we have selected several metrics:

- **Accuracy** - The ratio of correctly and incorrectly classified data samples
- **Precision** - tp / (tp + fp) (also called positive predicted value)
- **Recall** - tp/ (tp + fn) (also called sensitivity)
- **F1-score** - Harmonic mean of precision and recall

4 Task 1 - Congressional voting

4.1 Task description

The goal of this task is to predict the party of the politician based on his vote in congress.

4.2 Dataset description

A part of dataset could be seen in Fig. 1. Dataset consist of 218 rows and 16 attributes (columns ID and class are not attributes). Datasets contains only nominal (categorical, binary) data ('y'/'n'). Dataset also contains missing values ('unknown'). In the target value there are two possible classes - *republican*, *democrat* So the problem is binary classification. The distribution of the target value in training dataset could be seen in the Fig. 2. The number of non-missing values from each attribute could be seen in Fig.3.

	ID	class	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	religious-groups-in-schools	anti-satellite-test-ban	aid-to-nicaraguan-contras	mx-missile	immigration	synfuels-crporation-cutback	education-spending	superfund-right-to-sue	crime	duty-free-exports	export-administration-act-south-africa
0	67	republican	n	y	n	y	y	y	y	n	n	n	y	y	y	y	n	y
1	338	democrat	y	n	y	n	n	n	y	y	y	n	n	n	n	n	y	y
2	35	democrat	y	y	y	n	n	n	y	y	y	n	n	n	n	n	y	y
3	122	republican	n	unknown	n	y	y	y	n	n	n	y	n	y	y	y	n	y
4	420	democrat	y	y	y	n	n	n	y	y	y	n	n	n	n	n	n	y
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
213	91	democrat	y	n	y	n	n	n	y	y	y	y	n	n	n	n	y	y
214	15	republican	n	y	n	y	y	y	n	n	n	n	n	y	unknown	unknown	n	unknown
215	412	democrat	y	n	y	n	n	y	y	y	y	y	y	n	n	n	n	y
216	313	democrat	y	y	y	n	n	n	y	y	y	y	n	n	n	n	y	y
217	414	republican	y	y	y	y	y	y	y	y	n	y	unknown	unknown	unknown	y	n	y

Fig. 1: Part of the dataset

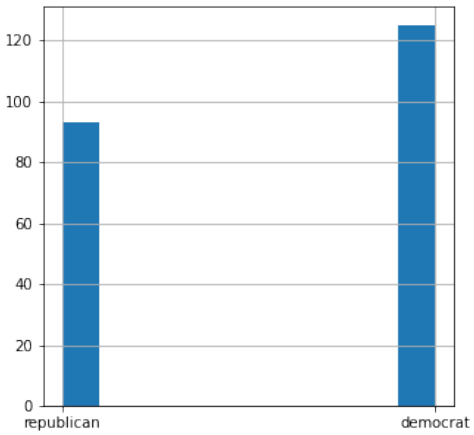

Fig. 2: Distribution of target value in training dataset

#	Column	Non-Null Count
0	handicapped-infants	212 non-null
1	water-project-cost-sharing	198 non-null
2	adoption-of-the-budget-resolution	214 non-null
3	physician-fee-freeze	213 non-null
4	el-salvador-aid	212 non-null
5	religious-groups-in-schools	211 non-null
6	anti-satellite-test-ban	209 non-null
7	aid-to-nicaraguan-contras	211 non-null
8	mx-missile	204 non-null
9	immigration	215 non-null
10	synfuels-crporation-cutback	206 non-null
11	education-spending	202 non-null
12	superfund-right-to-sue	205 non-null
13	crime	210 non-null
14	duty-free-exports	203 non-null
15	export-administration-act-south-africa	164 non-null

Fig. 3: Number of known values in dataset

4.3 Dataset preprocessing

Since there are not numbered data in the dataset, and there are a lot´s of missing values, the prep-processing would be necessary.

4.3.1 1. Possible prep-processing - label substitution

One possible prep-processing would be to substitute all *'y'* and *'n'* with number values 1 and 0. Then they are some missing values. The dataset is very small it is not good idea to throw any rows. In several articles/papers/tutorials, authors usually replace missing categorical values with the most common value in columns. It does not appear like a good idea here, because it can influence a voting results a lot so instead of that the unknown value will be considered as the third label. In the end, the labels ['y', "n', 'unknown'] would be replaced by [2,1,0]

4.3.2 2. Possible prep-processing - feature selection

Another possible prep-processing would be to do feature selection. This prep-processing step would be combined with the previous one. In the Figures. 6 and 5 are distributions of Democrats and Republikans votes. From these figures could be seen that each party has they have an almost common view on some of the voting subject and the other party has the opposite view (for example: *physican-fee-freeze, aid to nicaraguan-contrass...*, so from these attributes the party could be recognized, on the other hand, there are some attributes such as: *Water project cost sharing*, *Religious groups in school* and *immigration*, from witch it is not so easy to recognize the party. Feature selection could be also preformed with help of correlation matrix (Fig. 6). From this matrix could be seen that there are 2 attributes that does not correlate with the others - attribute with id 2 and 10, that are attributes named *Water project cost sharing*, *immigration*. So these two attribute would be removed

4.3.3 3. Possible prep-processing - Under/Over sampling

When preparing the classification training data, the datasets should be balanced (they should have similar number of datasamples in each class, but from the Fig. 2. could be seen that target values are not distributed equally, so there it would be possible to use oversampling (randomly duplicate data points in the minority class) or the undersampling (Randomly delete data points in the majority class). Since the dataset is already small, oversampling appears to be the better option.

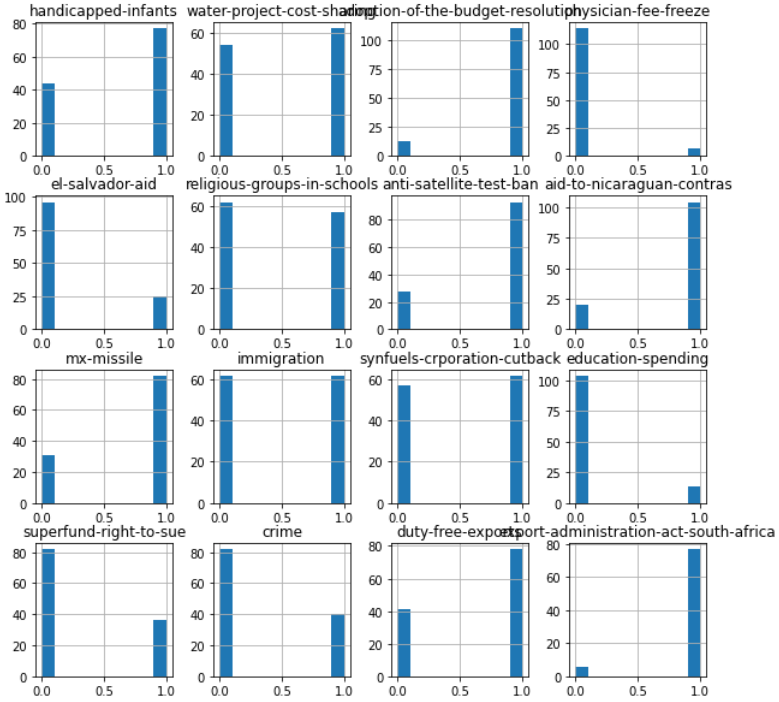

Fig. 4: Distributions of Democrats votes

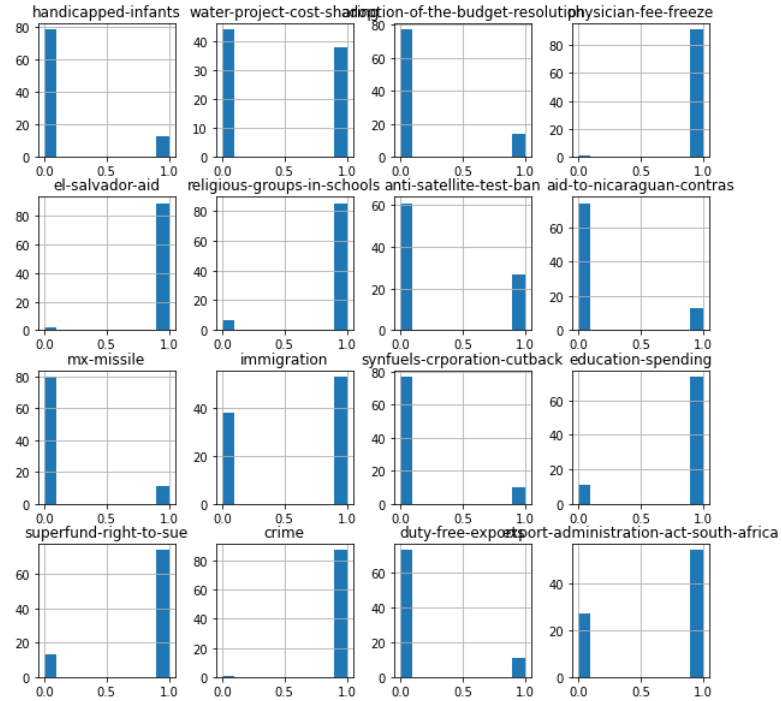

Fig. 5: Distributions of Republicans votes

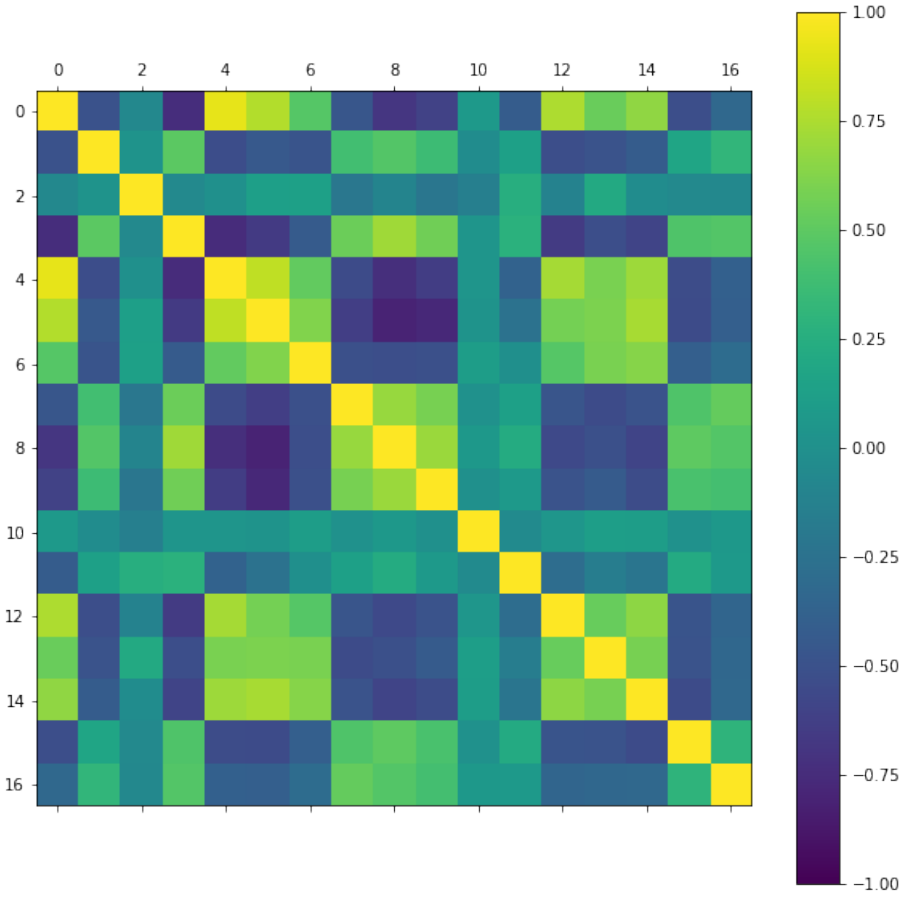

Fig. 6: Correlation matrix - numbers on axis represent the ID of attribute, where 0 is ID of class

4.4 Building models

Since the dataset is already very small it is not good idea to split it into train and test. The best option would be to use cross-validation to predict the performance of the model on real data.

4.4.1 Gaussian Naive Bayes (NB)

There are no hyper-parameters to tune. The results achieved by using this model are described in section 4.5

4.4.2 Random Forest Classifier (RFC)

Hyper-parameters used to tune the Random Forest Classifier are described in section 2. The data were prep-processed only by label substitution. By running several experiments (in the way where 1 parameter was changing and the others were fixed), the influence of hyper-parameters to model performance were examined. It was realized that parameter *max_depth* has usually some optimal value, where the model shows the best performance. When the

parameter *n_estimators* is changed, the performance of the model usually very oscillate, so it is hard to find some optimal value. It can not be said that parameters *max_features* and *criterion* has some the best values, because their influence on the performance very depend on the combination of previous parameters. All of these conclusion could be seen from the Fig. 7. In this case the recall metric usually has the highest values. By using grid search and cross-validation techniques, the parameters with the highest validation accuracy were tuned. The combination of parameters, that achieved the best performance is in the following table:

Parameter	max_depth	n_estimators	criterion	max_features
Value	5	20	'gini'	'sqrt'

Table 1: RFC parameters to have the best performance on Congressional voting dataset

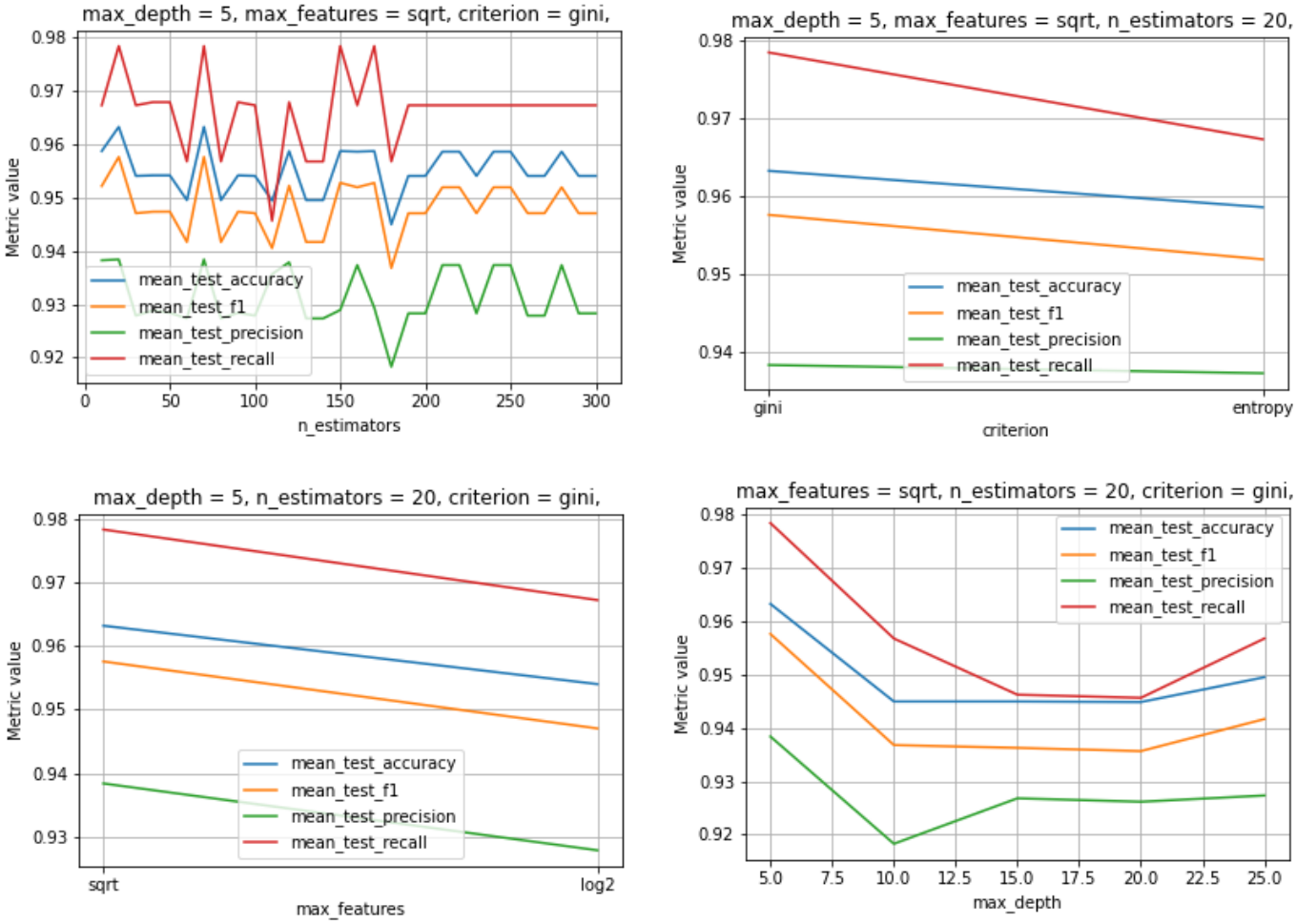

Fig. 7:

Fig. 8: Experiments with RFC hyper-parameters on Congressional voting dataset

4.4.3 Support Vector Machine (SVC)

Parameters used to tune the SVC classifier are described in the section 2. The data were prep-processed only by label substitution. By running multiple experiments (in the way that all parameters except for one, were set to specific value and the one was changing) it was realizes that *sigmoid* and rnf kernels tends to have better metric values than the polynomial kernel. From the experiments could be also seen that *gamma* parameter usually have some optimal value, where the model achieve the best results and influence to results from parameter C very depends on the combination of previous two parameters. These conclusions could be clear from Fig. 10. From the Fig. 10 - middle line left could be seen an interesting behaviour of *precision-metric*. All other metrics are decreasing their values except for this one. Using Grid search and cross-validation technique the best parameters to get the best validation accuracy were found. They could be seen in the following table:

Parameter	kernel	C	gamma
Value	sigmoid	20	78

Table 2: SVC parameters to have the best performance on Congressional voting dataset

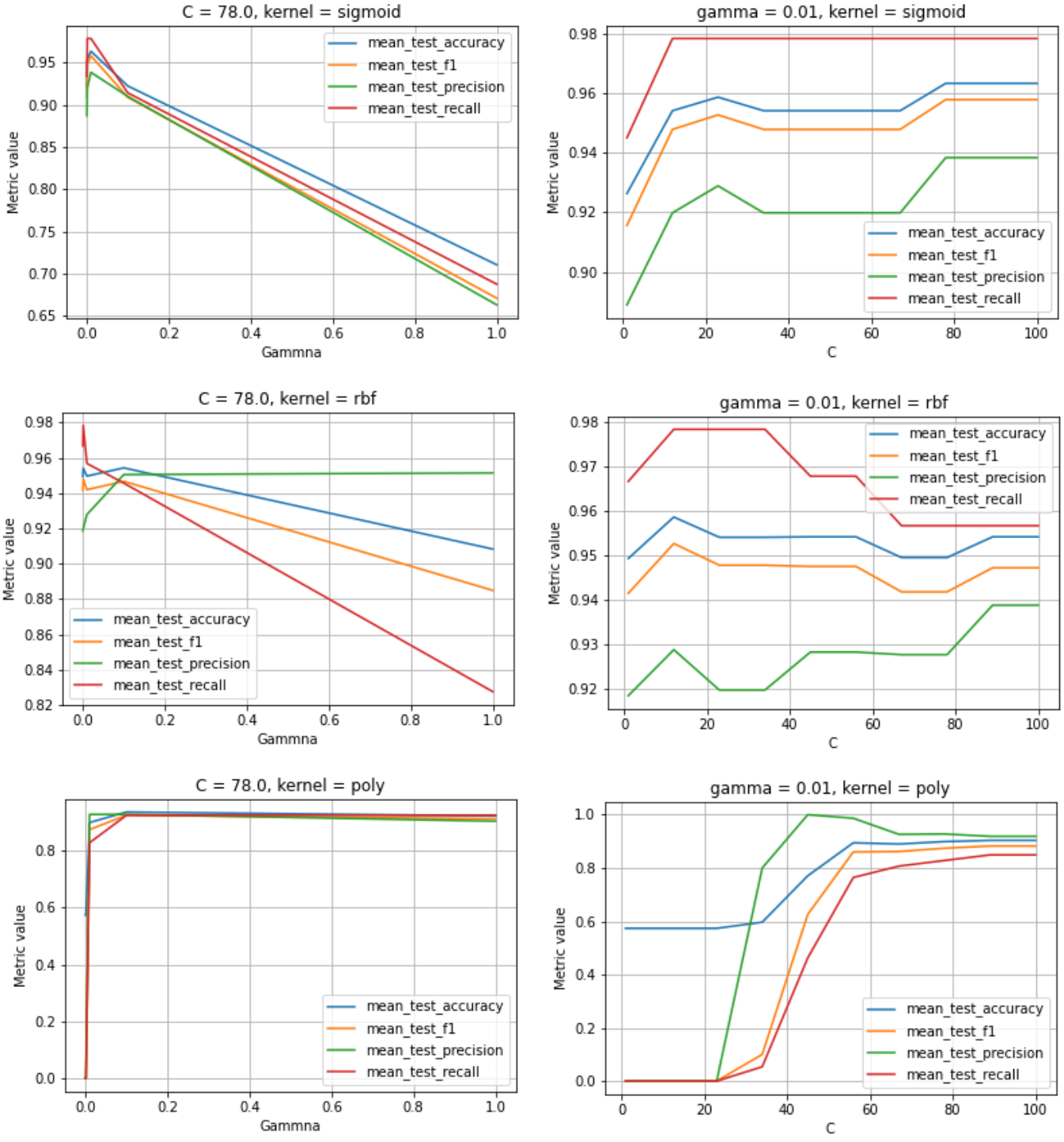

Fig. 9: The values of different metrics with fixed kernel and C (left), and with fixed gamma and kernel (left) - using only prep-processing described in 4.3.1 (All the metric values are mean values after finishing the cross-validation

4.5 Results

In this section all tree models would be compared. The important values could be seen in the table 3 and 4. From the table 3 could be seen that is was possible to achieve very similar results by using all three models (In this table

the results by using only 1. prep-processing). The weakest performance could be seen in Naive Bayes model, but on the other hand, it is very simple and easy to implement and there are no hyper-parameters, so there is no need to find some optimal values, so it will save some (computational) time. In the table 2 could be seen the comparison of models and it´s metrics with using different prep-processing methods. To get this results, the models were tuned to parameters where they performed the best results (described in previous chapters) and this result were get by using cross-validation

Model	Training time [s]	Cross-valid accuracy	Test accuracy(kaggle)
NB	0.00299	0.9540	0.87692
RFC	0.0026 - mean per set of params	0.9540	0.95384
SVC	0.147- mean per set of params	0.9493	0.95384

Table 3: Comparison of models

Preprocessing	Model [s]	Accuracy	F1	Recall	Precision
1	NB	0.9540	0.9470	0.9672	0.92826
	RFC	0.9493	0.9410	0.94619	0.9256
	SVC	0.9540	0.9470	0.9282	0.9672
1+2	NB	0.9520	0.9523	0.9282	0.9600
	RFC	0.9279	0.9448	0.9359	0.9083
	SVC	0.9520	0.9523	0.9600	0.9483
1+3	NB	0.9540	0.9470	0.9672	0.9282
	RFC	0.9586	0.9570	.9251	0.9015
	SVC	0.9495	0.9424	0.9188	0.9678

Table 4: Comparison of models preprocessings

5 2. Task - Location

5.1 Task description

The task is to predict a geosocial type for users based on their location data

5.2 Dataset description

A part of dataset could be seen in the Fig. 10. Datasets consist of 4000 data samples and 446 attributes. All values are only nominal binary data (1/0). In the dataset there are no missing values. Firstly, with these information we know that there is no need for the scaling as a prep-processing step. The target value consist of 30 labels. The distribution of these labels could be seen in histogram in Fig. 11.

	class	1	2	3	4	5	6	7	8	9	...	437	438	439	440	441	442	443	444	445	446
0	11	0	0	0	1	1	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
1	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	9	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
3	8	0	0	0	1	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
4	3	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3995	2	0	0	0	1	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
3996	27	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	1	0	0	0	0
3997	15	0	0	0	1	0	0	0	0	1	...	0	1	0	0	0	0	0	0	0	0
3998	19	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3999	25	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	1	0	1	1

Fig. 10: Part of the dataset

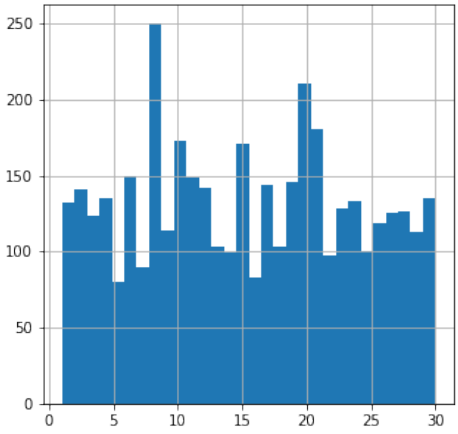

Fig. 11: Histogram of distribution of target value

5.3 Pre-processing

5.3.1 1. possible prep-processing - under/over sampling

The target values, should be distributed equally, but they are not, so over/under sampling techniques would be needed. In this prep-processing step would be choosed threshold value (120). Classes that are presented in more samples would be reduced. Classes that has less amount of datasamples than this threshold would be enlarge. The histogram of classes after applying this process could be seen in Fig. 12.

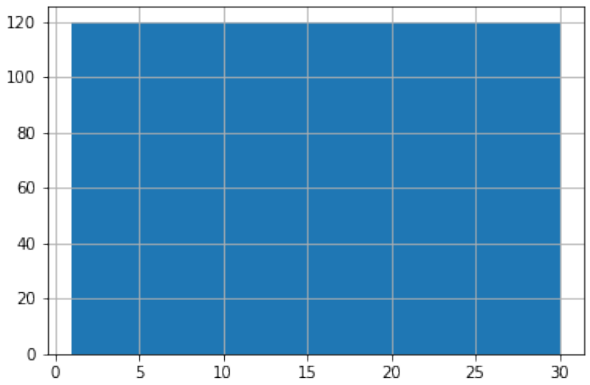

Fig. 12: Histogram of distribution of target value by applying under and over sampling with threshold = 120

5.3.2 2. possible prep-processing - Principal component analysis (PCA)

This specific dataset is high dimensional so as a prep-processing step we should try to find a subset of features to represent the same level of information in the dataset. For this purpose, dimension reduction techniques are needed. Dimension reduction techniques are also used for identify outliers, once we reduce the dimensions of our dataset and represent our data with fewer features it is easier to see the outliers. PCA is one of the most popular Dimension reduction techniques, in order to use it, all the features of the dataset should be on the same scale. Otherwise, a feature with large values would dominate the results.

5.4 Building models

5.4.1 Gaussian Naive Bayes

After implementing PCA with 50 components on train and test set, we have calculated the accuracy of this method as : 0.6129

5.4.2 Support Vector Machines

Both prep-processing steps and several combination of parameters were used to get the best performance of the model (with help of Grid search technique). The results could be seen in the following table:

MODEL	Prep-processing	C	gamma	kernel	Accuracy (kaggle)
SVM	PCA (3 comp)	1	0.1	'rbf'	0.1885
	PCA (5 comp)	0.5	1	'linear'	0.4937
	PCA (50 comp)	10	0.01	'rbf'	0.6923
	Under/Over	3	0.01	'rbf'	0.68734

Table 5: Comparison

5.4.3 Random Forest Classifier

This model were fit with the help of grid search technique. After the model were fit. All of the features from the dataset were examined by their importance. The result could be seen in the Fig. 13 and Fig. 14. Then the fist 5O most important features were used to get the results. (The best achieved results are in the table 6)

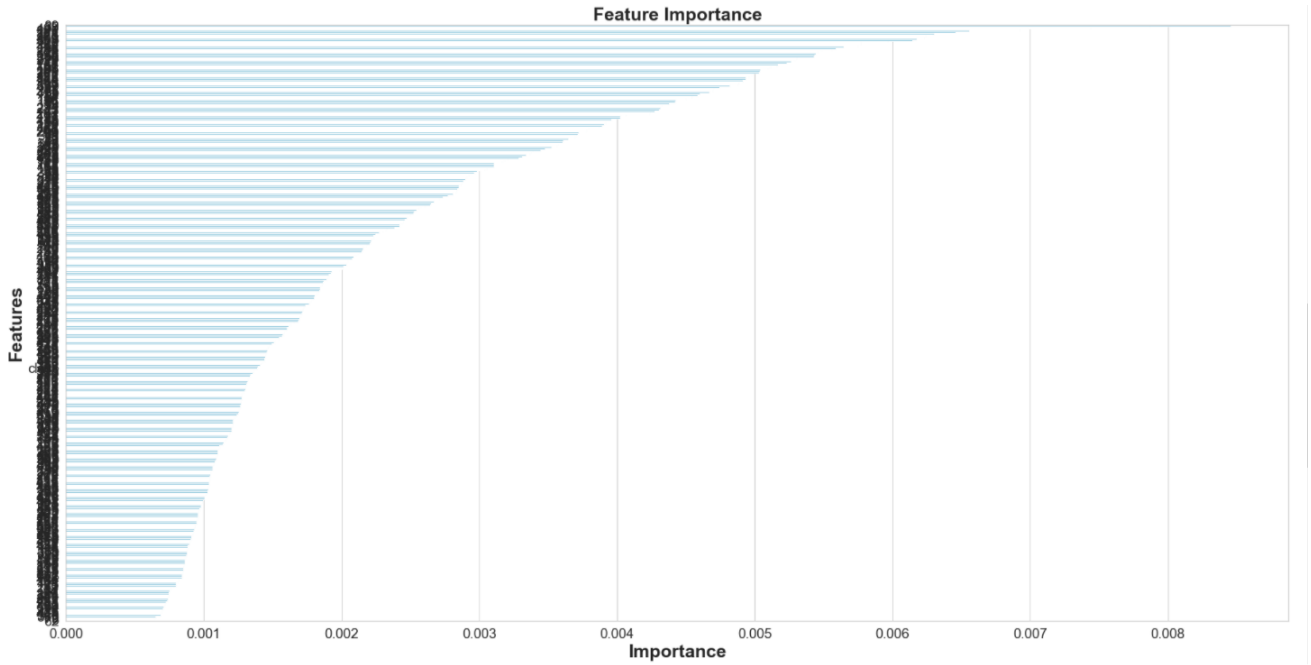

Fig. 13: Distribution of feature importance

	Features	Gini-Importance
0	39	0.008455
1	35	0.007867
2	423	0.007438
3	181	0.006924
4	295	0.006559
5	131	0.006457
6	368	0.006304
7	228	0.006292
8	163	0.006245
9	409	0.006229

Fig. 14: Feature importance

Prep-processing	n-estimators	criterion	max_depth	max_features)	Accuracy(kaggle)
feature selection	50	'gini'	None	'auto'	0.5756

Table 6: Results from RFC with feature selection

5.5 Results

The best obtained results could be seen in the following table. From the table could be seen that the best results were achieved with Support vector machine model with using PCA prep-processing with 50 components.

Model	Prep-prosessing	Accuracy(kaggle)
NB	PCA(50)	0.6129
SVC	PCA (50)	0.6923
RFC	Feature selection	0.5756

Table 7: Results from RFC with feature selection

6 3. Task - Rain in Austria

6.1 Dataset description

A part of dataset could be seen in Figure 15. Dataset consist of 145460 rows and 23 columns. 22 of these columns are attributes and the column *RainTomorrow* is our target. Dataset contains 7 categorical [*Date*, *Location*, *WindGustDir*, *WindDir*9*am*, *WindDir*3*pm*, *RainToday*, *RainTomorrow*] and 16 numerical [*MinTemp*, *MaxTemp*, *Rainfall*, *Evaporation*, *Sunshine*, *WindGustSpeed*, *WindSpeed*9*am*, *WindSpeed*3*pm*, *Humidity*9*am*, *Humidity*3*pm*, *Pressure*9*am*, *Pressure*3*pm*, *Cloud*9*am*, *Cloud*3*pm*, *Temp*9*am*, *Temp*3*pm*] columns. Unique values of the columns, total count of missing values in the columns and percentage of the missing values in the columns can be seen in Figure 16.

	C23	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
1	RainTomorrow	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3p
2	No	2008-12-01	Albury	13.4	22.9	0.6	NA	NA	W	44	W	WNW	20	24	71	22
3	No	2008-12-02	Albury	7.4	25.1	0	NA	NA	WNW	44	NNW	WSW	4	22	44	25
4	No	2008-12-03	Albury	12.9	25.7	0	NA	NA	WSW	46	W	WSW	19	26	38	30
5	No	2008-12-04	Albury	9.2	28	0	NA	NA	NE	24	SE	E	11	9	45	16
6	No	2008-12-05	Albury	17.5	32.3	1	NA	NA	W	41	ENE	NW	7	20	82	33
7	No	2008-12-06	Albury	14.6	29.7	0.2	NA	NA	WNW	56	W	W	19	24	55	23
8	No	2008-12-07	Albury	14.3	25	0	NA	NA	W	50	SW	W	20	24	49	19
9	No	2008-12-08	Albury	7.7	26.7	0	NA	NA	W	35	SSE	W	6	17	48	19
10	Yes	2008-12-09	Albury	9.7	31.9	0	NA	NA	NNW	80	SE	NW	7	28	42	9
11	No	2008-12-10	Albury	13.1	30.1	1.4	NA	NA	W	28	S	SSE	15	11	58	27
12	Yes	2008-12-11	Albury	13.4	30.4	0	NA	NA	N	30	SSE	ESE	17	6	48	22
13	Yes	2008-12-12	Albury	15.9	21.7	2.2	NA	NA	NNE	31	NE	ENE	15	13	89	91
14	Yes	2008-12-13	Albury	15.9	18.6	15.6	NA	NA	W	61	NNW	NNW	28	28	76	93
15	No	2008-12-14	Albury	12.6	21	3.6	NA	NA	SW	44	W	SSW	24	20	65	43
16	NA	2008-12-15	Albury	8.4	24.6	0	NA	NA	NA	NA	S	WNW	4	30	57	32
17	No	2008-12-16	Albury	9.8	27.7	NA	NA	NA	WNW	50	NA	WNW	NA	22	50	28

Fig. 15: part of the dataset

```
UNIQUE VALUES IN COLUMNS
Date             3436
Location           49
MinTemp           389
MaxTemp           505
Rainfall          681
Evaporation       358
Sunshine          145
WindGustDir        16
WindGustSpeed      67
WindDir9am         16
WindDir3pm         16
WindSpeed9am       43
WindSpeed3pm       44
Humidity9am       101
Humidity3pm       101
Pressure9am       546
Pressure3pm       549
Cloud9am           10
Cloud3pm           10
Temp9am           441
Temp3pm           502
RainToday           2
RainTomorrow        2
dtype: int64

COUNT OF MISSING VALUES IN COLUMNS
Date                0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
Pressure3pm     15028
Cloud9am        55888
Cloud3pm        59358
Temp9am          1767
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64

PERCENTAGE OF MISSING VALUES IN COLUMNS
Date            0.000000
Location        0.000000
MinTemp         0.010209
MaxTemp         0.008669
Rainfall        0.022419
Evaporation     0.431665
Sunshine        0.480098
WindGustDir     0.070989
WindGustSpeed   0.070555
WindDir9am      0.072639
WindDir3pm      0.029066
WindSpeed9am    0.012148
WindSpeed3pm    0.021050
Humidity9am     0.018246
Humidity3pm     0.030984
Pressure9am     0.103568
Pressure3pm     0.103314
Cloud9am        0.384216
Cloud3pm        0.408071
Temp9am         0.012148
Temp3pm         0.024811
RainToday       0.022419
RainTomorrow    0.022460
dtype: float64
```

Fig. 16: Information about dataset

6.2 Dataset preprocessing

We have missing values, categorical columns and unscaled data in our dataset. Preprocessing is needed.

6.2.1 Dealing with missing values

Firstly,when we look at Figure 10. We can see that the percentage of null values is really high in columns 'Cloud9am' :0.38, 'Sunshine': 0.48, 'Evaporation' : 0.43, Cloud3pm : 0.40, the others are so much lower in percentage than these attributes.That is why we considered dropping these columns. Comparison of effect on accuracy and learning time between removing and not removing these columns (we have calculated only with RandomForestClassifier with holdout method) can be seen in Figure 17. We decided to keep those columns. For all the other missing values, a

```
accuracy score when columns are dropped
              precision    recall  f1-score   support

           0       0.88      0.95      0.91     26448
           1       0.76      0.52      0.62      7430

    accuracy                           0.86     33878
   macro avg       0.82      0.74      0.77     33878
weighted avg       0.85      0.86      0.85     33878

accuracy score when columns are dropped:  0.8587283782986008
learning time: 10.674997806549072


accuracy score when columns are NOT dropped
              precision    recall  f1-score   support

           0       0.88      0.96      0.92     13158
           1       0.78      0.55      0.64      3768

    accuracy                           0.86     16926
   macro avg       0.83      0.75      0.78     16926
weighted avg       0.86      0.86      0.85     16926

accuracy score when columns are NOT dropped:  0.8639962188349285
learning time: 4.940783739089966
```

Fig. 17: Should we remove the columns with high percentage of null values : NO !

possible prep-processing step would be to separate categorical and numerical columns and then replace the missing values in the categorical ones with mode of that column and replace the ones in the numerical columns with median of that column. But we have not used these prep-processing step because we dealt with the whole data in the prep-processing and in order to avoid 'Data Leakage' we have splitted our dataset right before we set the model, and scaled with a pipeline function while its fitting the dataset into the model. Instead we removed all the rows that contained missing values and after that we still had 56420 rows.

6.2.2 Heatmap- highly correlated columns

Another possible prep-processing step would be to eliminate one of the two highly correlated columns that we see from the heatmap Figure 18 . Columns 'MinTemp' and 'Temp9am' has 0.91 correlation,columns 'MaxTemp' and 'Temp3pm' has 0.98 correlation, columns 'Pressure9am' and 'Pressure3pm' has 0.96 correlation. So we drop 'Temp9am','Temp3pm' and 'Pressure9am'and compare the accuracy and run time again (only for RainForestClassifier with holdout method). The accuracy 0.8639 (from Figure 17) drops to 0.8595 but it shortens the learning time by 0.12 s. We decide to not to drop those columns as well.

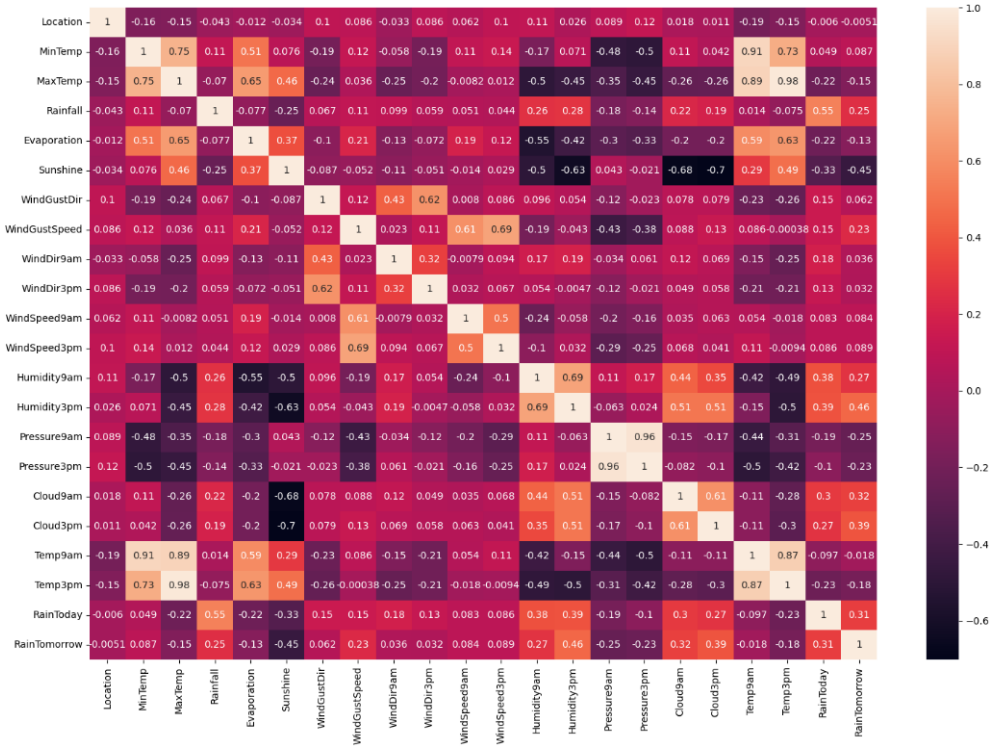

Fig. 18: Heatmap of whole dataset

6.2.3 Encoding the categorical variables

We have considered two methods for encoding:One-hot encoding and Label encoding and tried both of them on RainForestClassifier with holdout method. As we can see from 19, label encoding gave better accuracy and run-time. So, we used label encoder.

```
accuracy score with one-hot-encoding  0.859269762495569
learning time with one-hot-encoding: 5.265928268432617
accuracy score with label encoding  0.8635235732009926
learning time with label encoding: 4.845041990280151
```

Fig. 19: comparison between encoding methods: winner:'LaberEncoder()'

6.2.4 Scaling

We did the scaling right after the split. Using the Standart Scaler, that removes the mean from data and scale them to unit variance.

6.2.5 Outliers removing

After split and scaling. We looked at our train and test data and as you can see from 20 our train data (as well as our test data) has outliers and if we eliminate them we get a graph like in Figure 21 and our accurancy increases from 0.8637 to 0.8651.

Now, that we have prepped the data we can move on to models:

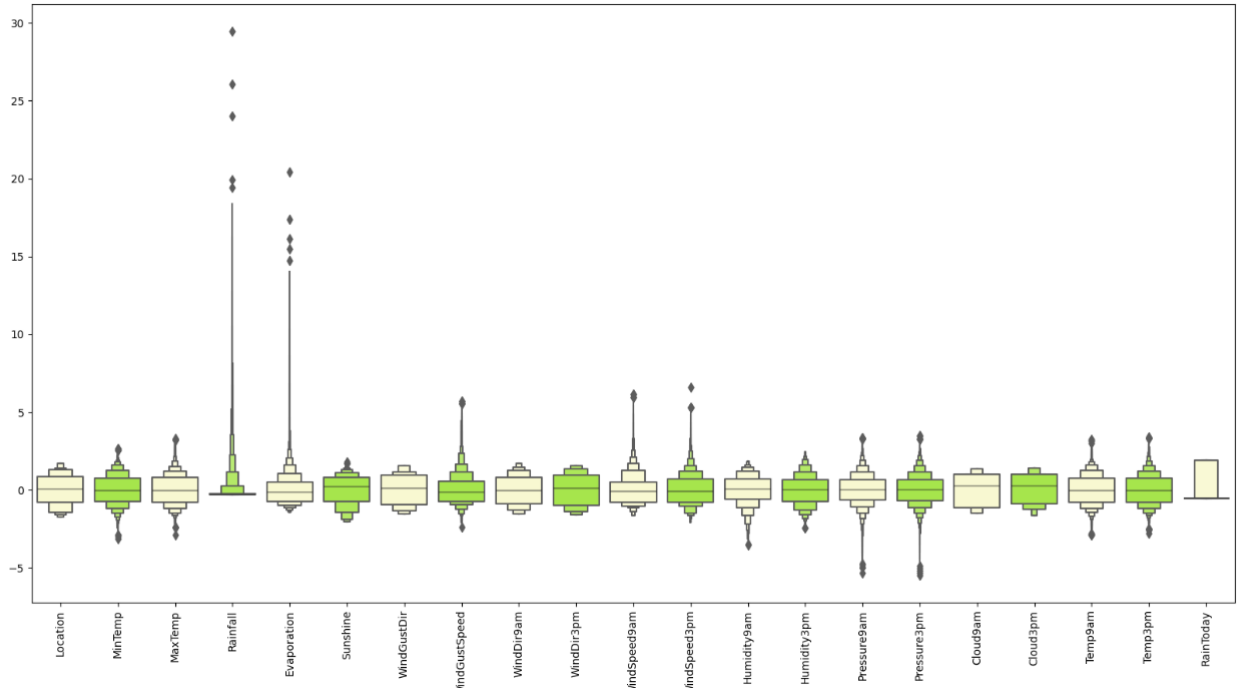

Fig. 20: Detecting Outliers

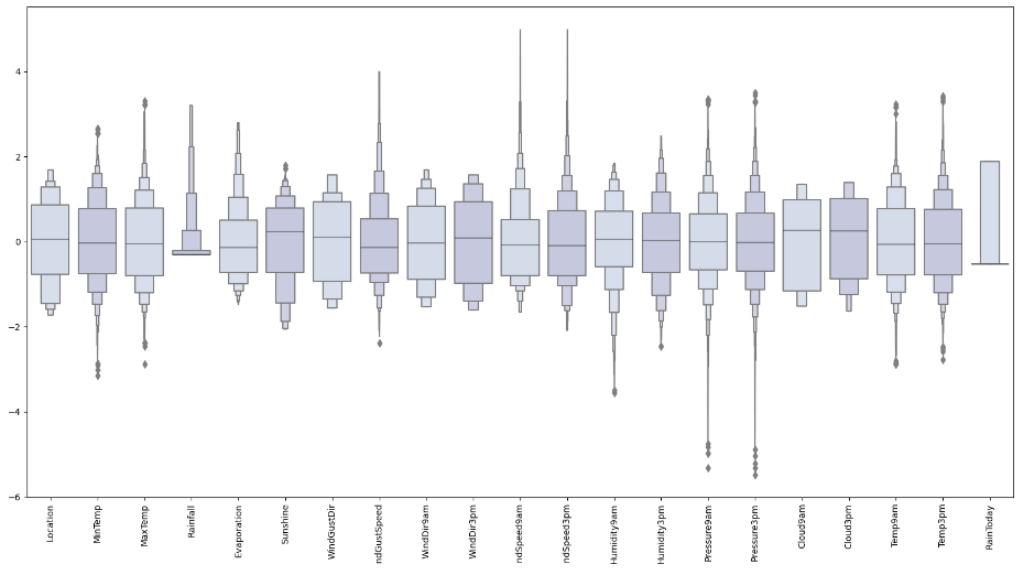

Fig. 21: After removing outliers

6.3 Classifier Models

We have chosen Gaussian Naive Bayes, Random Forest Classifier and k-NN models for this dataset. We have considered Support Vector Machines as well and applied SVM on this prepped data set with holdout method accuracy score for SVM holdout method : 0.8659 learning time for SVM holdout method : 31.0270 But when we started to search for best parameters through gridsearch, we realized that SVM is not a suitable model for this specific data set. The time that needed to accomplish gridsearch successfully was too long. That is why we chose our third model

as k-NN. **Holdout method** Holdout Method is the simplest sort of method to evaluate a classifier. In this method, the data set is separated into two sets, called the Training set and Test set. Model performs function of assigning data items in a given target class. It is very straight forward.

6.3.1 Gaussian Naive Bayes

We can not look at the hyperparameters on this model.

6.3.2 RandomForestClassifier

hyperparameters for example: best parameters for RFC in our dataset :(cv=5) ('criterion': 'entropy', 'max depth': 10, 'max features': 'auto', 'n estimators': 200)

6.3.3 LightGBMClassifier

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages;Faster training speed and higher efficiency,lower memory usage,Better accuracy,support of parallel and GPU learning,capable of handling large-scale data. for example: Best parameters for LightGBM in our dataset : (cv=2) ('learning rate': 0.1, 'maxdepth': 10, 'min child samples': 15, 'numleaves': 60, 'regalpha': 0.03)

6.4 Comparison

Comparison between our models can be seen from Tables 8,9 and 10.

Models	Training time[s]	Accuracy	f1 score	recall	Precision
NB	0.0372s	0.8014	0.81	0.80	0.83
RFC	6.04038s	0.8660	0.86	0.87	0.86
LightGBM	0.6721s	0.8703	0.87	0.87	0.86

Table 8: Comparison of models -Holdout

Cross-valid count	Models	cross-valid score	Standart deviation of score	Accuracy with best parameters
cv=2	NB	0.7900	0.0400	0.8000
	RFC	0.8500	0.0100	0.8604
	LightGBM	0.8400	0.0200	0.8690
cv=5	NB	0.8000	0.0300	0.8000
	RFC	0.8400	0.0100	0.8591
	LightGBM	0.8100	0.0600	0.8677
cv=10	NB	0.8000	0.0500	0.8000
	RFC	0.8400	0.0200	0.8612
	LightGBM	0.7800	0.1000	0.8677

Table 9: Comparison of models

Models	Accuracy	f1 score	recall	Precision
NB	0.801488	0.81	0.80	0.83
RFC	0.859181	0.85	0.86	0.85
LightGBM	0.869018	0.86	0.87	0.86

Table 10: Comparison of models -cv gridsearch

7 4. Task - Red Wine quality

7.1 Task

The task is to predict the quality of Red Wine by know Red Wine features.

7.2 Dataset description

A part of dataset could be seen in Figure 22. Dataset consist of total 1599 data samples with 11 attributes. The attributes are Interval and Ratio data types. The distribution of values of all attributes could be seen in Fig. 23. There are no missing values. The target value consist of 6 possible labels for quality (numbers 3 to 8). The distribution of all target values could be seen in Fig. 24.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

Fig. 22: A piece of dataset

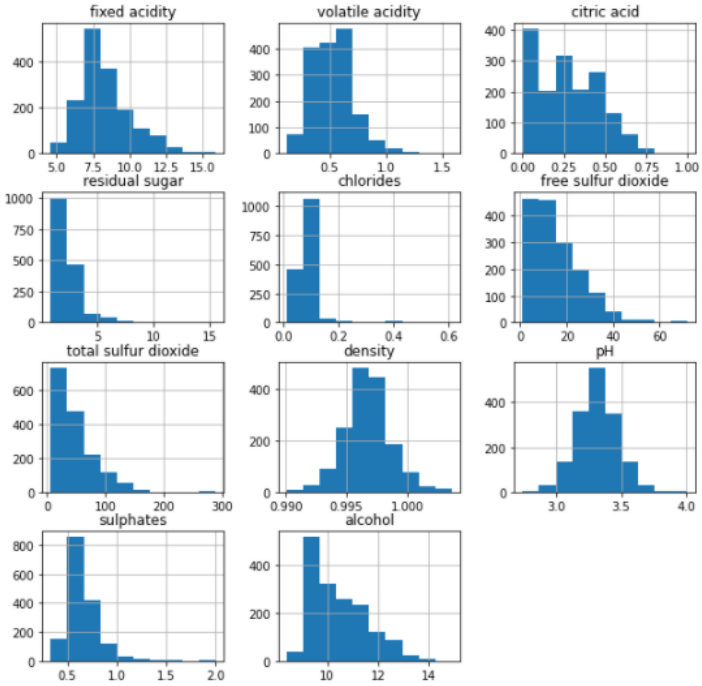

Fig. 23: Distribution of values in dataset

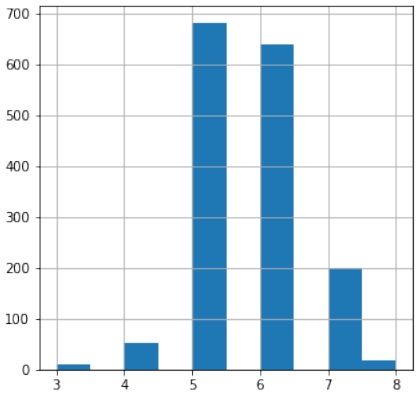

Fig. 24: Target value distribution

7.3 Dataset prep-processing

7.3.1 1. Possible prep-processing - Scaling

Since we have numeric values in the dataset in various ranges, one option would be to scale the values in ranges 0-1. Before applying this step it is necessary to split the data into test and train set first, than do the scaling on train sets, and after apply the scaling constants to test set. From the distribution of data in Fig. 23 could be seen that the data does not have unit variance, and different mean, so the best option appears to use the Standart Scaler. However in this exercise the min-max scaler is used, because the Standart Scaler was already used in different dataset.

7.3.2 2. Possible prep-processing - Re-sampling

From the Fig. 24 could be seen that the data samples are very non-equally distributed in the classes. So one option would be to to create just 2 classes - good quality/bad quality. Classes that has label 5 and less would be bad quality and the others good quality. Bad quality would have label 0 and good quality label 1.

7.4 Building models

7.4.1 Gaussian Naive Bayes

The result obtained from Gaussian Naive Bayes could be found in tables 13, 14 and 15.

7.4.2 Random Forest Classifier

This model were build by using only prep-processing described in section 7.3.2. By doing experiments with Random Forest Classifier hyperparameters. Parameters that are examined in random forest are described in section 2. By changing parameters (usually 3 of them were fixed and 1 was changing) and after evaluating the model by using cross-validation. It was realized that parameter max_depth has some optimal value, where the model has the best performance. By changing parameter $n_estimators$ the metrics value usually very oscillating, so it is very hard to find some optimal value. Models had better performance when the parameter $max_features$ were set to *log* and also when parameter *criterion* were set to *gini* . These conclusions could be clear from graphs in the Fig. 25. From this Figure could be also seen that Percision has highest values. By using the grid search technique the parameters where found the parameters the model performed the best accuracy. The parameters are following in the following table:

Parameter	max_depth	n_estimators	criterion	max_features
Value	15	180	'gini'	'log2'

Table 11: RFC parameters to have the best performance on Red Wine quality dataset

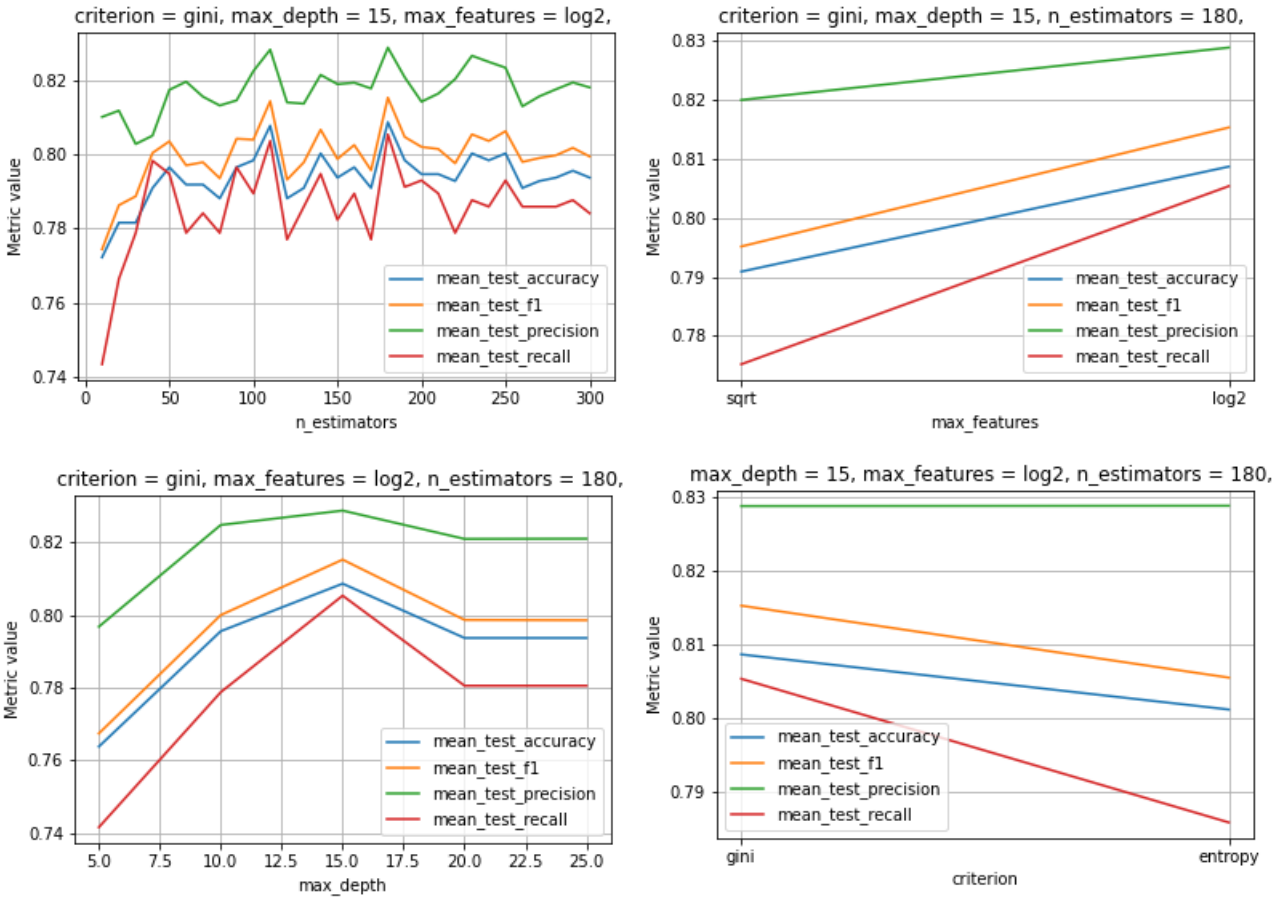

Fig. 25: Influence of models parameters to model performance (RFC)

7.4.3 Support Vector Machine

By running several experiment with SVC model it was realized that here the model performance very depends on the choose of parameter *kernel*. The other parameters does not have a really big influence on the performace. This conclusions could be seen from Fig. 26. The model with the best performance appears to be the model with combination of parameters, that are in the following table:

Parameter	C	gamma	kernel
Value	700	0.001	'rbf'

Table 12: SVC parameters to have the best performance on Red Wine quality dataset

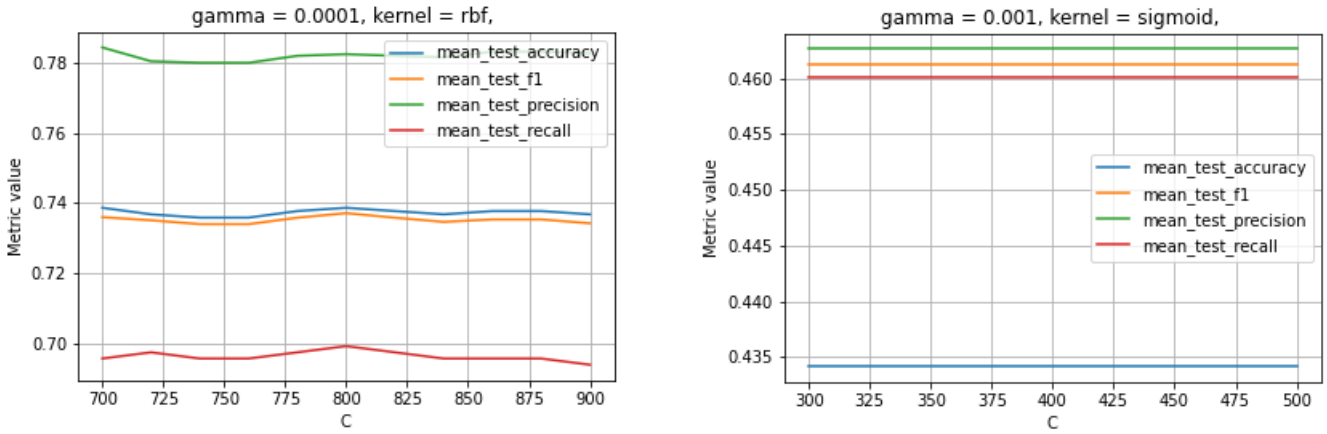

Fig. 26: Influence of models parameters to model performance (SVC model)

7.5 Comparison of models

The validation accuracy and the test accuracy (also the training time) of models with the best tuned parameters are in the table 7. From the table could be seen that the best accuracy on the test set was achieved by Random Forest Classifier model.

Model	Training time [s]	Cros-valid accuracy	Test accuracy
NB	0.0029	0.7320	0.7424
RFC	0.300 - mean per set of params	0.7927	0.8087
SVC	0.273- mean per set of params	0.7946	0.7973

Table 13: Comparison of models

7.6 Influence of prep-processing

In this section would be compared the influence of prep-processing steps. All parameters of all models would be fixed in the values that were found above by using the Grid search technique (the one with the best performance). Only the prep-processing would change. The results could be seen in the table 8. From the table could be seen that if we add scaling as an prep-processing step (prep-processing 1+2) it has big influence only on the Random forest classifier model, where the test accuracy, and also precision and f1 decreases.

	preprocessing 2							
	valid_acc	test_acc	valid_recall	test_recall	valid_precision	test_precision	valid_f1	test_f1
NB	0.7320	0.7424	0.7646	0.7601	0.7381	0.7758	0.7505	0.7679
RFC	0.7927	0.8087	0.7964	0.8098	0.8218	0.8517	0.8006	0.8302
SVC	0.79460	0.7973	0.78938	0.8000	0.8163	0.8413	0.8035	0.8201
	preprocessing 1 + 2							
	valid_acc	test_acc	valid_recall	test_recall	valid_precision	test_precision	valid_f1	test_f1
NB	0.7264	0.7386	0.7628	07567	0.7313	0.7724	0.7461	0.7645
RFC	0.7927	0.6420	0.7858	0.7327	0.8238	0.5482	0.8024	0.6272
SVC	0.7936	0.8030	0.7876	0.8163	0.8139	0.8275	0.8016	0.8219

Table 14: Comparison of all metric with different prep-processing steps

7.7 Comparison of holdout and cross-validation

In this section the holdout and cross-validation is compared. All parameters of all models would be fixed in the highest performance values, and prep-processing 1+2 is applied to the data. From the table could be seen that cross-validation gave worse prediction for RFC model- precision and f1 metric.

	Cross-validation				Cross-validation - Test			
	accuracy	recall	precision	f1	accuracy	recall	precision	f1
NB	0.7264	0.7628	0.7313	0.7461	0.7386	0.7567	0.7724	0.7645
RFC	0.7927	0.7858	0.8238	0.8024	0.6420	0.7327	0.5482	0.6272
SVC	0.7936	0.7876	0.8139	0.8016	0.8030	0.8163	0.8275	0.6272
	Holdout				Holdout-test			
	accuracy	recall	precision	f1	accuracy	recall	precision	f1
NB	0.7627	0.7787	0.7719	0.7753	0.7386	0.7567	0.7724	0.7645
RFC	0.8418	0.8846	0.8070	0.8440	0.7803	0.7979	0.8034	0.8006
SVC	0.8232	0.8518	0.8070	0.8288	0.8011	0.8135	0.8275	0.8205

Table 15: Comparison of Cross-validation and Holdout

8 Summary

In this exercise 4 classification task were performed. On these dataset 3 models were examined - Naive Bayes, Random Forest and Support Vector Machine. From all things that were realized during solving this task, it could be conclude, that Naive Bayes has the weakest performance (that was expected) the performance of each of other model very depends on the hyper-parameters and on prep-processing.