Akhmetzhanov Ravil, RO - 21
Machine Learning (F21)
Innopolis University, 2021
Report on Assignment 1

1. Introduction

This assignment is devoted to solving the task of flight delay estimation using machine learning. It's a real task from the industry because the dataset comes from Innopolis University partner company analyzing flights delays. The main parts of this report:

- ➔ Task and data description
- ➔ Data preprocessing
- ➔ Outliers detection and removal
- ➔ Comparison of selected machine learning models
- ➔ Conclusion

2. Task and data description

The task is to estimate the time of flight delay (in minutes), so it's a regression task. The given dataset is shown in figure 1:

	Depature Airport	Scheduled depature time	Destination Airport	Scheduled arrival time	Delay
0	SVO	2015-10-27 07:40:00	HAV	2015-10-27 20:45:00	0.0
1	SVO	2015-10-27 09:50:00	JFK	2015-10-27 20:35:00	2.0
2	SVO	2015-10-27 10:45:00	MIA	2015-10-27 23:35:00	0.0
3	SVO	2015-10-27 12:30:00	LAX	2015-10-28 01:20:00	0.0
4	OTP	2015-10-27 14:15:00	SVO	2015-10-27 16:40:00	9.0
...	...	...	...	...	...
675508	SVO	2018-08-31 23:50:00	SVX	2018-09-01 02:10:00	0.0
675509	LED	2018-08-31 23:50:00	SVO	2018-09-01 01:10:00	0.0
675510	SVO	2018-08-31 23:55:00	EGO	2018-09-01 01:20:00	0.0
675511	SVO	2018-08-31 23:55:00	TSE	2018-09-01 03:15:00	0.0
675512	SVO	2018-08-31 17:25:00	IKT	2018-08-31 23:05:00	379.0

675513 rows × 5 columns

Figure 1. Dataset for flight delay estimation

Each entry in the dataset file corresponds to a flight and the data was recorded over a period of 4 years. There are 4 predictors and 1 target (‘Delay’). All of the predictors are objects (shown in Figure 2), so to get numerical values it’s necessary to preprocess them.

```
types = dataset.dtypes
print("Number categorical featues:", sum(types=='object'))
print(types)
```

```
Number categorical featues: 4
Depature Airport            object
Scheduled depature time     object
Destination Airport         object
Scheduled arrival time      object
Delay                      float64
dtype: object
```

Figure 2. Types of data presented in the dataset

3. Data preprocessing

To obtain data from categorical features it’s necessary to use an appropriate encoding. Because there are 179 unique airports (shown in Figure 3), the choice for encoder - **LabelEncoder**. One Hot Encoder wasn’t used because in our case it would increase data dimensionality too much (177 + 179 = 356 extra dimensions).

```
print(dataset['Destination Airport'].nunique())
print(dataset['Depature Airport'].nunique())
```

```
177
179
```

Figure 3. Number of unique airports

But Label Encoder has its disadvantage - labels are ordered randomly (in the existing order of the data), which can add noise while assigning an unexpected order between labels. Our model can think, that SVO > OTP, and that adds some error.

The next step was **extracting new features** from 'Scheduled departure time'. ‘Departure Month’, ‘Departure Day of the Week’, and ‘Flight Duration’ were obtained.

- Delay may depend on the month because it definitely depends on the weather, and the weather is different every month.
- The author of this report made a suggestion, that delay may depend on the day of the week because on weekends pilots might be more relaxed than from Monday till Friday.
- Flight Duration may affect delay too.

To get all these predictors from the original dataset ‘Scheduled departure time' and ‘Scheduled arrival time' were transformed to (datetime64[ns]) format. But later these predictors were dropped and were not used to train ML models.

It’s very important to get rid of NaNs and use **imputers**. But in the given dataset there are no empty cells or cells with NaNs, so no imputting is needed. Scaling can improve model’s performance, so **RobustScaler** was used.

After this preprocessing dataset was split and print (Figure 4). For simplicity ‘Flight Duration’ was printed against the target variable.

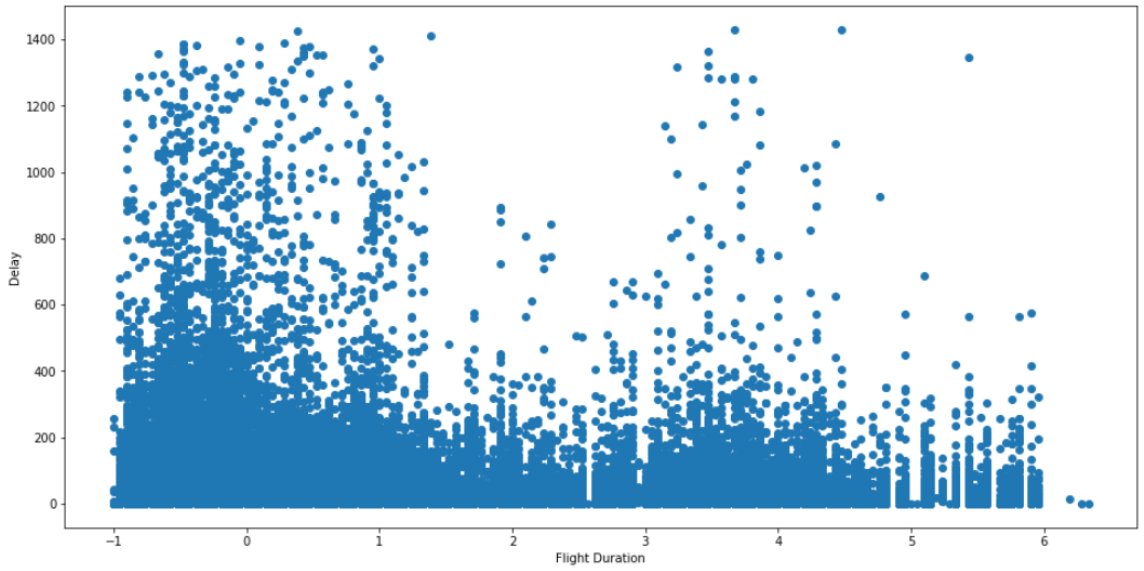

Figure 4. Delay dependency on Flight Duration

As you can see on Figure 4, there are a very small number of big delays, and most delay a less than 1 hour. Value count (Figure 5) showes us that there are 391070 samples with 0 delay, which is 57,9% from the original dataset.

```
dataset['Delay'].value_counts()
```

```
0.0       391070
1.0        38963
2.0        28729
3.0        22501
4.0        17167
           ...
706.0          1
1284.0         1
1042.0         1
1037.0         1
1002.0         1
Name: Delay, Length: 1075, dtype: int64
```

Figure 5. Value count of the dataset

4. Outlier detection and Removal

Outliers are observation points that are distant from other observations. Outliers can either be a mistake or just variance. Firstly, it's necessary to find them. In this experiment, Z-score method was used. Z - score is the number of standard deviations (***σ***) by which the value of a raw score is above or below the mean value of what is being observed or measured. Using Z-score we make an assumption, that data have a normal distribution (Figure 6).

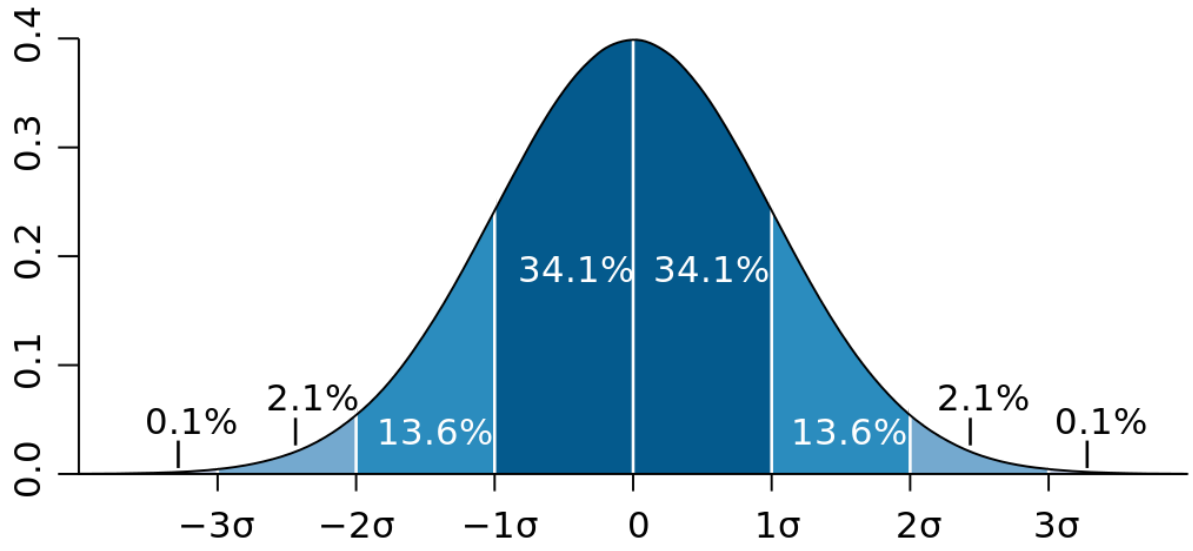

Figure 6. The plot of normal distribution

These data points which are way too far from the mean will be treated as the outliers. In most cases a threshold of |3| is used i.e if the Z-score value is greater than or less than |3| respectively, that data point will be identified as outliers.

Firstly we need to apply this outlier detection algorithm to a small fraction of data, to identify, do outliers ever exist in the dataset. Figure 6 shows, that there are 941 outliers in the first 15000 samples.

```
from scipy import stats

#cheching for 1 month (approximately 2% = 15000 samples)
threshold = 3
sample = train_data[0:15000]
z = np.abs(stats.zscore(sample))
out_indexes = np.where(z > threshold)

unique_indexes = np.unique(out_indexes[0])
print(f' Number of outliers: {len(unique_indexes)}')
sample.shape
```

```
 Number of outliers: 941
(15000, 6)
```

Figure 6. Outliers detection in [0 : 15000] samples

That means, that outliers are present in the dataset. This approach was applied to the whole dataset, outliers were removed and cleaned dataset was plotted (Figure 7). A total number of outliers is 29531.

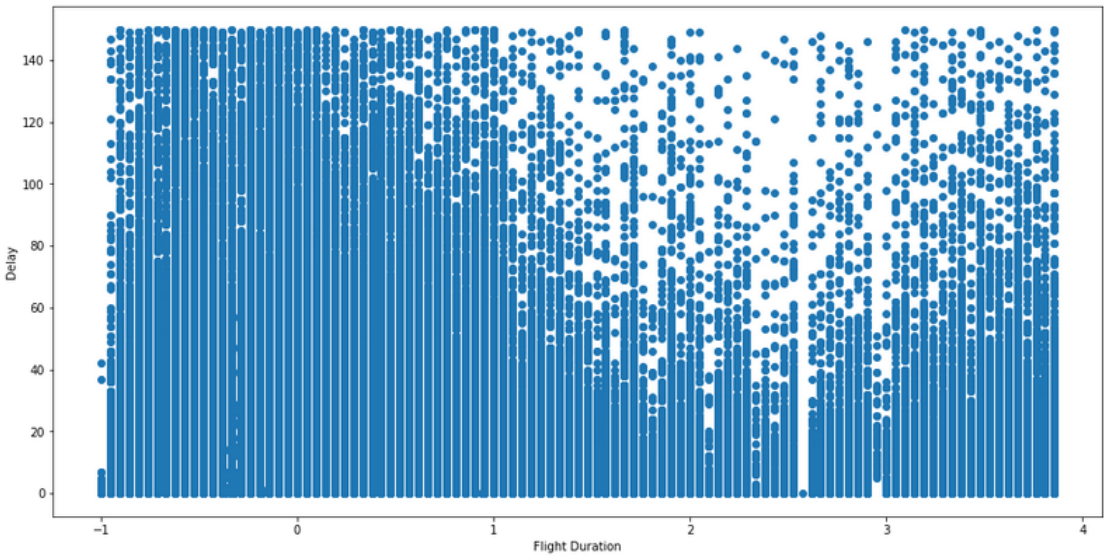

Figure 7. Data plot after outlier removal

X train dataset is shown in Figure 8. As you can see, there are 5 predictors left. ML models will be trained using this data.

	Depature Airport	Destination Airport	Departure Month	Departure Day of the Week	Flight Duration
4	-0.508197	0.000000	10	1	-0.142857
5	-1.491803	0.000000	10	1	0.047619
7	-1.786885	0.000000	10	1	1.714286
8	0.000000	0.450000	10	1	3.285714
9	0.196721	0.000000	10	1	0.857143
...	...	...	...	...	...
499052	0.000000	-1.400000	12	6	3.714286
499053	0.000000	-1.283333	12	6	0.619048
499054	0.000000	-2.116667	12	6	3.523810
499055	0.000000	-1.316667	12	6	0.666667
499057	0.000000	-0.816667	12	6	3.476190

479092 rows × 5 columns

Figure 8. X train dataset

5. Comparison of selected machine learning models

Next step after preprocessing is applying machine learning models. Because it's a regression task and we can use only models that were mentioned in this course, so our choice is:

1. Linear regression
2. Linear regression with regularization (Lasso was used)
3. Polinomial regression

Models' performance is shown in Tables 1 and 2.

Table 1. Models' performance on train data

Model	Linear regression	Lasso regression	Polynomial regression
Root MSE	17.05	17.05	17
MAE	9.35	9.37	9.29
R2 score	0.0093	0.00915	0.016

Table 2. Models' performance on test data

Model	Linear regression	Lasso regression	Polynomial regression
Root MSE	39.97	39.97	39.89
MAE	11.32	11.33	11.12
R2 score	0.0031	0.0029	0.0073

Important notion: Root MSE and MAE show us errors in minutes. Maximum possible R2 score is 1 - it means functional relation between predictors and target. Minimal R2 score = 0. Minimum appropriate R2 score is 0.5.

Linear regression: This model has a big test error in predicting the delaybecause our data are highly non linear, which is seen in Figure 4. Even train error is high and R2 score is very low, so it's definitely underfit.

Lasso regression: As we know, Ridge regularization is preferable when there are many predictors influencing the response. But there are only 5, so Lasso was chosen.

Lasso has hyperparameter λ, and it was tuned by cross-validation. It was implemented using GridSearchCV from sklearn.model_selection, because it's the simplest way to do it. The best $\lambda = 0.1$.

In Tables 1 and 2 you can see, that this model didn't improve the performance and all metrics are almost the same. So - underfit.

Polynomial regression: Polynomial degree in this model needs to be tuned. Polynomial regression with degree 3 gives us the best result among all models (this is shown in code). Metrics for this model are in Tables 1 and 2.

Every model has approximately same metrics on train data. But polynomial regression has the best performance on test data. Root MSE and MAE are almost the same, and R2 score is higher than in previous models, but it's still too low. This model is also underfiittng.

Conclusion

Despite doing all necessary preprocessing, removing outliers, and using 3 different machine learning models, a small prediction error wasn't achieved. All 3 models are underfitting. There are several reasons for it:

1) Lack of information in the original dataset. There were only departure and arrival time and airports. Some important information about the

weather, experience of the crew is missing. Probably, weather conditions influence flight delays the most. So, **the first idea for improving the results** is:
Using a given route and time we can find information about the weather in the internet.

2) Too simple models for complex and non linear data were used. Linear, Lasso, and Polynomial regression just couldn't fit such data. **The second idea for improving the results** - using more complex model.