Assignment 1
Flight Delay Forecasting

ay.ahmetshin@innopolis.ru

September 2021

1 Introduction

In this report we obtain different approaches-models to estimate flight **Delay**. We will use :

1. Linear models
2. Polynomial models

and also try to prove that regularisation helps us in developing such models.

2 Technical stack

1. **visualization:** matplolib, seaborn, basemap
2. **data manipulation:** pandas, numpy
3. **modeling:** sklearn

3 Dataset

Each entry in the dataset file corresponds to a flight and the data was recorded over a period of 4 years. These flights are described according to 5 variables. A sneck peek of the dataset can be seen in the table below:

	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

Figure 1: Initial Dataset

Also provide info, we can pay attention that we do not have null values, and in preprocessing phase we should not to use Imputing and take care about values:

Dataframe dimensions: (675513, 5)

	Depature Airport	Scheduled depature time	Destination Airport	Scheduled arrival time	Delay
column type	object	object	object	object	float64
null values (nb)	0	0	0	0	0
null values (%)	0	0	0	0	0

Figure 2: Types and Description

```
Data columns (total 5 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Depature Airport        675513 non-null  object
 1   Scheduled depature time  675513 non-null  object
 2   Destination Airport     675513 non-null  object
 3   Scheduled arrival time  675513 non-null  object
 4   Delay                   675513 non-null  float64
dtypes: float64(1), object(4)
```

Figure 3: Info

Also we should pay attention to our main feature - **Delay** in whole data set we have 391070 **Zero-Delay**, it is a 57.9%

4 Data Preprocessing and Visualization

In first step we should extend our data set and add new features that will help us to analyze outliers and encode categorical data properly.

4.1 Extend Features

Calculate duration of the flight, extract time departure, convert all object to concrete type or category data, new description of dataset below:

	Depature Airport	Scheduled depature time	Destination Airport	Scheduled arrival time	Delay	Duration	Year	Month	Day	Departure time	IS_delay
0	SVO	2015-10-27 07:40:00	HAV	2015-10-27 20:45:00	0.0	785	2015	10	27	07:40:00	0
1	SVO	2015-10-27 09:50:00	JFK	2015-10-27 20:35:00	2.0	645	2015	10	27	09:50:00	1
2	SVO	2015-10-27 10:45:00	MIA	2015-10-27 23:35:00	0.0	770	2015	10	27	10:45:00	0
3	SVO	2015-10-27 12:30:00	LAX	2015-10-28 01:20:00	0.0	770	2015	10	27	12:30:00	0
4	OTP	2015-10-27 14:15:00	SVO	2015-10-27 16:40:00	9.0	145	2015	10	27	14:15:00	1

Figure 4: New dataset

4.2 Encoding

Use LabelEncoding() on categorical data: 'Departure time', 'Departure Airport', 'Destination Airport'. After it use correlation matrix between features in our dataset: Here we cant obtain that some features related to each other more.

Figure 5: Correlation matrix

Above, we have assigned a label to each airport. The correspondence between the label and the original identifier has been saved in the **label airport** list. Also we can show another correlation from this extended dataset:

4.3 Outliers

To detect and remove outliers we used z-score and also remove data depend on some correlation plot between two main features 'Duration' and 'Delay'. We paid attention to small amount of big duration and small delays. Because in

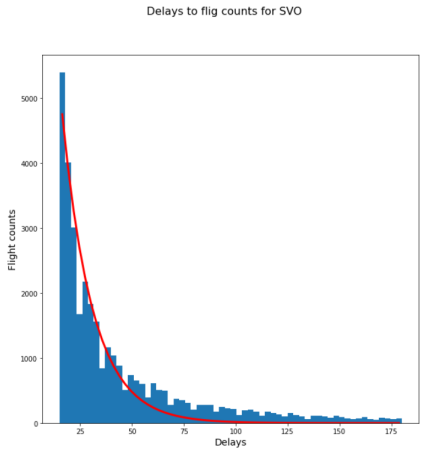


Figure 6: SVO departure airport

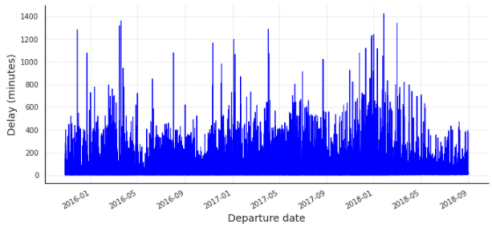


Figure 7: Delay distribution

this dataset there is a interesting trend small flight duration have more amount of big delays than long duration flights.

Z-score: simple way to remove outliers. Also we can do it using interquartile range (IQR) and other ways to smooth dataset.

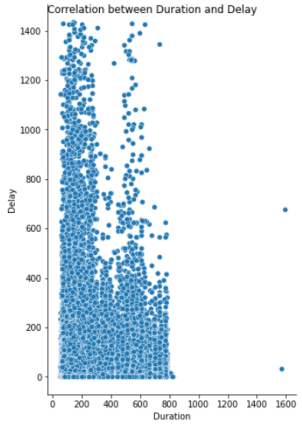
$$Z_{score} = \frac{df_i - mean}{std} \tag{1}$$

5 Linear Regression

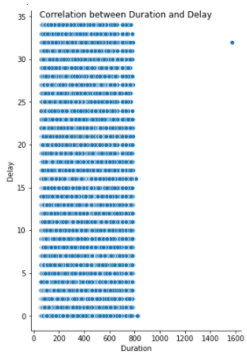
Simple linear regression is an approach for predicting a quantitative response using a single feature (or "predictor" or "input variable") It takes the following form:

$$y = \beta_0 + \beta_1 x \tag{2}$$

β_0 is the intercept, β_1 is the coefficient for x



(a) Before removing



(b) After removing

Figure 8: Train dataset

5.1 Results

MSE score of the fit. In practice, we can have a feeling of the quality of the fit by considering the number of predictions where the differences with real values is greater than 10-15 minutes:

```
lm = linear_model.LinearRegression()
model = lm.fit(train['Duration'].to_numpy().reshape(-1, 1), train['Delay'])
predictions = lm.predict(train['Duration'].to_numpy().reshape(-1, 1))
print("MSE =", metrics.mean_squared_error(predictions, train['Delay']))
```

```
MSE = 44.968572850783
```

Figure 9: Train

```
lm = linear_model.LinearRegression()
model = lm.fit(train['Duration'].to_numpy().reshape(-1, 1), train['Delay'])
predictions = lm.predict(test['Duration'].to_numpy().reshape(-1, 1))
print("MSE_test =", metrics.mean_squared_error(predictions, test['Delay']))
```

```
MSE_test = 1606.5331876267144
```

Figure 10: Train

6 Polynomial Regression

Here we see that the fit is particularly bad with a MSE ¿ 1000 (the exact value depends on the run and on the splitting of the dataset), which means that the fit performs poorly when generalizing to other data.

In practice, this can be explained by the fact that during the separation in train and test sets, data with no equivalent in the training set was put in the

test data. Moreover we can pay attention that we have a hige number of zero delays about 58% in whole dataset and when we splitting data depend on years we see that in Test part is 79.60% also have zero delay. Thus, when calculating the prediction, the model has to perform an extrapolation. If the coefficients of the fit are large (which is often the case when overfitting), extrapolated values will show important values, as in the present case. In order to have a control over this phenomenon, we can use a regularization method

```
poly = PolynomialFeatures(degree = 4)
regr = linear_model.LinearRegression()
X_ = poly.fit_transform(train['Duration'].to_numpy().reshape(-1, 1))
regr.fit(X_, train['Delay'])
result = regr.predict(X_)
print("MSE_train =", metrics.mean_squared_error(result, train['Delay']))
```

```
MSE_train = 44.94298288713152
```

Figure 11: Train

```
X_ = poly.fit_transform(test['Duration'].to_numpy().reshape(-1, 1))
result = regr.predict(X_)
score = metrics.mean_squared_error(result, test['Delay'])
print("Mean squared error = ", score)
```

```
Mean squared error =  1606.3847413592114
```

Figure 12: Test

7 Regularization method

linear regression works by selecting coefficients for each independent variable that minimizes a loss function. However, if the coefficients are too large, it can lead to model over-fitting on the training dataset. Such a model will not generalize well on the unseen data. To overcome this shortcoming, we do regularization which penalizes large coefficients.

Ridge regression is an extension of linear regression where the loss function is modified to minimize the complexity of the model. This modification is done by adding a penalty parameter that is equivalent to the square of the magnitude of the coefficients.

$$Lossfunction = OLS + \alpha * summation(squaredcoefficientvalues) \quad (3)$$

where OLS - Ordinary least squares

In the above loss function, alpha is the penalty parameter we need to select. Using an l1 norm constraint forces some weight values to zero to allow other coefficients to take non-zero values.

```
X_ = poly.fit_transform(test['Duration'].to_numpy().reshape(-1, 1))
result = ridgereg.predict(X_)
score = metrics.mean_squared_error(result, test['Delay'])
print("Mean squared error withy regurilization = ", score)
```

```
Mean squared error withy regurilization =  1606.8035571768091
```

Figure 13: with Regularization

8 Conclusion

Two problems of this dataset: 1) huge amount of zero-delays 2) small amount of additional features for correct predictions. Nevertheless we can extended our dataset and and some practical features as: 'weather', 'flight id', 'team experience' that can give us more correlations for predicting delay. Also we can improve and compare different types of regularization Lasso, Logistic Regression also we can apply SVR and obtain better results.

9 References

1. GitHub Repo
2. Regularization tricks
3. Flight Delay Estimation
4. Models description
5. Estimating linear models in delay problem