Task 3

As a final task, your supervisor assigned to you to investigate if it is possible to train a linear regression model that could predict the departure delay a flight may have by using, as input, its origin (column "ORIGIN"), its airways (column "CARRIER"), and its departure time (column "DEP_TIME"). Again, you should use Python and DataFrames, this time with MLlib. You should pay attention to transform the string-based input features using the proper representation format, and you should explain your choices. Special attention should be given to the "DEP_TIME" column: your supervisor told you that you should only consider the corresponding hour of the day (which, of course, should be transformed in a one-hot representation). For your training and testing workflow you should first remove outliers (see Task 2) and then split your dataset into two parts, one that will be used for training reasons and it will contain 70% of the entries, and a second one containing the remaining entries and which will be used for the assessment of the model. No need to implement a more sophisticated evaluation process (e.g., based on k-fold) is required in this phase.

For the task3 question the goal is to run a linear regression model in order to make predictions about the departure delay of a flight. We are going to use the DEP_DELAY as a depending variable and the variables ORIGING, CARRIER and DEP_TIME as predictors. Specifically, for the DEP_TIME variable we have to make special data manipulation techniques in order to construct a new variable with labels the ours of the day and the value "cancelled" which stands for nu values. Since we are going to use Spark to fit our regression model we have to one – hot encode our three input variables into vectors and then pipe them into the model to make the computations.

First of all, we are going to filter the null values in the DEP_DELAY column. (I do this operation from the beginning because when I fitted the model the system showed errors regarding null values in the DEP_DELAY column).

After this, we can estimate unique values for the first two input variables , ORIGIN has 360 labels and CARRIER has 17 respectively , both columns has string type . Later we will index them and then we are going to use one – hot encoding technique in order to construct a vectors with length of 360 and 17 respectively which will consists of 0 and 1 values in order to pipe these to the model as said.

For the DEP_TIME column the problem was that the original data was not in a timestamp format or a clear 4 -digit length in order to construct the time DEP_TIME_HOUR. Instructions of the assignment says ("you should only consider the corresponding hour of the day") so we have to take the first two digits of the column in order to have a complete 24 houred column format. The trick was used here is a transformation fill with leading zeros, this means that when there is an observation with 3 digits e.g 855 then a zero is filled in front of this 3digit observation transform it into an easier manipulated format, thus we now have 0855 for this observation, 0030 for a two digit '30" valued observation etc. Now that the column consists only with 4-digit entries, we create a new column called DEP_TIME_HOUR consisted of the first two digits which stand for the hour and entries labeled as "cancelled". This label replaced the "nu" entries that were cancelled flights. I decided not to drop this information as it will resulted in information loss.

Regarding the outliers, we do the same procedure as in task2, we calculate the 0.01 percentile of the total flights number. Again, the value of the 0.01 percentile is 82 and airports labeled as AKN, PGV, DLG and GST must be dropped.

Now after we cleaned our data, we are ready to make the one – hot transformations.

Since all three columns are of string we have to use the StringIndexer first and transform each label to an Index. Then we feed this output to the OneHotEncoder, which maps a categorical feature represented as a label index to a binary vector with at most a single one- value (0 ,1) indicating the presence of a specific feature from among the set.

Finally, we gather all these vectors together with VectorAssembler and transform it into a single vector column. We do this because we will pipe this vector in our regression model , fit the data into the model and run a prediction . Regression output below.

```
+------------------+---------+--------------------+
|        prediction|DEP_DELAY|            FEATURES|
+------------------+---------+--------------------+
|  1.662077040936441|    -12.0|(398,[135,362,373...|
| 6.2029013976246351|     -1.0|(398,[30,356,374]...|
| 7.5476460433307751|     -3.0|(398,[0,357,378],...|
|16.199949819281876|     -4.0|(398,[11,357,387]...|
|10.667151559112668|     24.0|(398,[25,356,377]...|
| 9.8177488896409121|     -1.0|(398,[38,356,378]...|
|13.237875107053956|     23.0|(398,[88,362,384]...|
|17.320161818246497|     11.0|(398,[2,358,387],...|
|12.132983747310586|    148.0|(398,[20,363,377]...|
|11.152252341891144|     -2.0|(398,[26,359,382]...|
+------------------+---------+--------------------+
only showing top 10 rows

R Squared (R2) on test data = 0.0439607
Root Mean Squared Error (RMSE) on test data = 48.0007
```

We conclude that the model performs very poor in terms of prediction. The R2 on test data is extremely low, only 4% and the RMSE is very high 48%. In order to improve the model, we need further training into the data using some training method like repeated cross validation or adding more predictors in order to increase the predictive ability of the model.