CEBD 1260 – Machine Learning

Project

***Predicting Insurance Quote Conversion Into Sale***

**A picture containing drawing

Description automatically generated**

Presented to

**Prof. Yimin Nie**

Presented by

**Julio Suni Alvarez**

*November 2020*

Concordia University

Table of Contents

[The Project 1](#_Toc57407774)

[The Dataset 1](#_Toc57407775)

[Data Processing and Exploratory Data Analysis 2](#_Toc57407776)

[Missing Data Analysis 2](#_Toc57407777)

[Type Check and Conversion 3](#_Toc57407778)

[Correlation Analysis and Dropping Features 4](#_Toc57407779)

[Feature Engineering 5](#_Toc57407780)

[Process Categorical in Depth 7](#_Toc57407781)

[Splitting Data 8](#_Toc57407782)

[LightGBM Model 9](#_Toc57407783)

[Conclusion 12](#_Toc57407784)

[Comments 12](#_Toc57407785)

[Bibliography 12](#_Toc57407786)

The Project

The following project is a machine learning model that will predict the probability of a quote to be converted into an actual sale.

These quotes are requested by customers interested in getting an insurance service. The insurance company gives a quotation to the customer and then the customer will decide if he will contract the insurance or not. This model will predict both whether the client will purchase the insurance or not in absolute terms, but also it will give the prediction based on a probability of the customer of acquire the insurance.

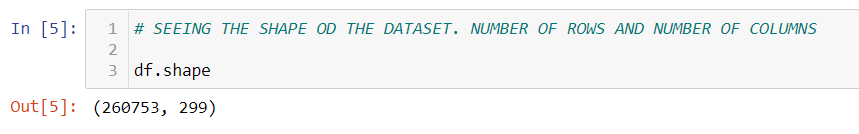
For the project we will use a dataset and we will apply python code and libraries such as pandas, numpy, matplotlib, scikit-learn and other libraries to perform analysis and build an algorithm and model that will do the predictions based on historical data of previous quotes information and an extra column (target) indicating whether the quote got converted into a sale or not.

In this this project, I will be performing supervised learning for the model. In that sense, I will analyze the features and will perform feature engineering for the model to learn from known features.

Finally, I will shot that the AUC for this project is roughly 0.97 which is a very good metrics for predictions.

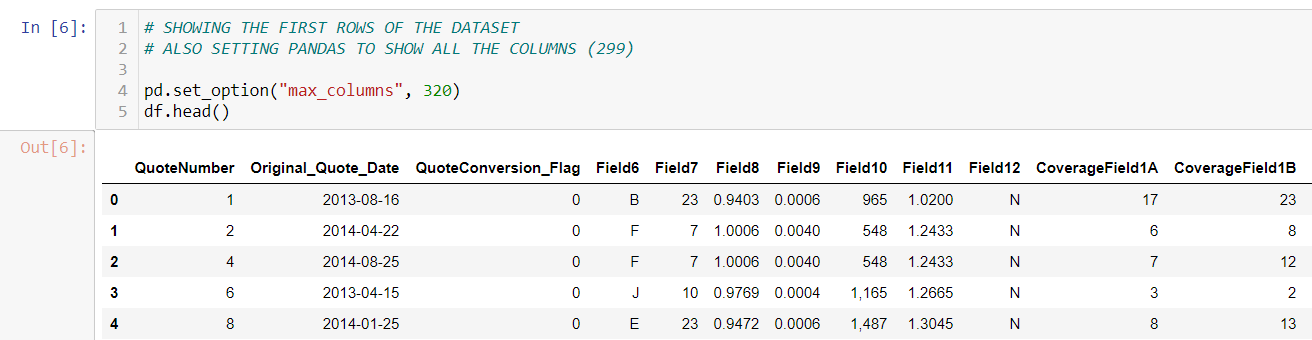
The Dataset

The dataset is CSV file of 260,753 rows and 299 columns.



The dataset also contains only 3 meaningful columns and the rest are of the format of: Field6, Field7, CoverageField1A, CoverageField1B, etc.

Here a glance of how the dataset looks like.

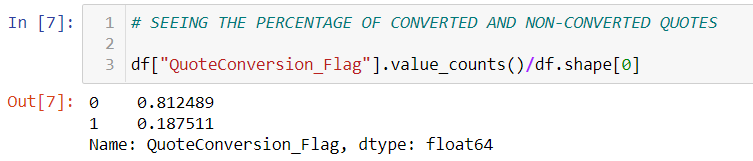


As it is shown, only QuoteNumber, Original\_Quote\_Date and QuoteConversion\_Flag are meaningful columns.

It is also clear that the target corresponds to the QuoteConversion\_Flag column.

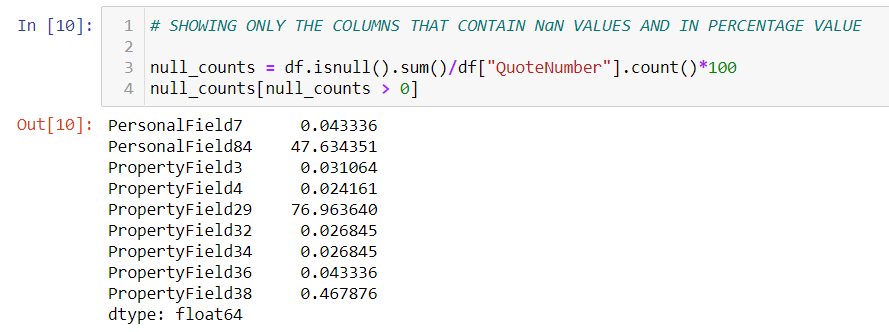
Data Processing and Exploratory Data Analysis

I first analyzed how imbalanced is the dataset in terms of number of record that were converted into sale and number of record that were not converted into a sale.

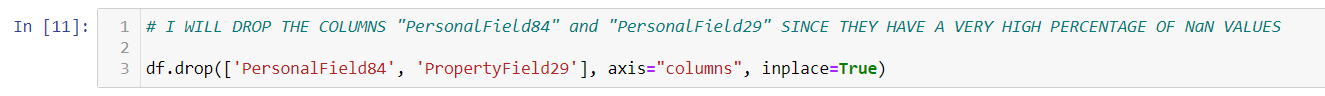


Missing Data Analysis

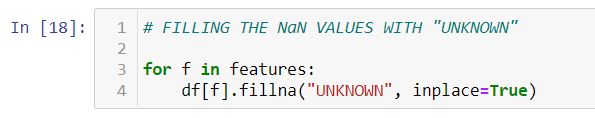
I analyzed which features had a high percentage of null values.



I dropped the columns with a high percentage of null values.

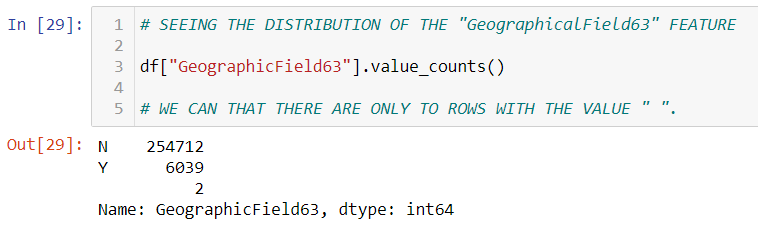


For the rest, I decided to fill the null values with “UNKNOWN”

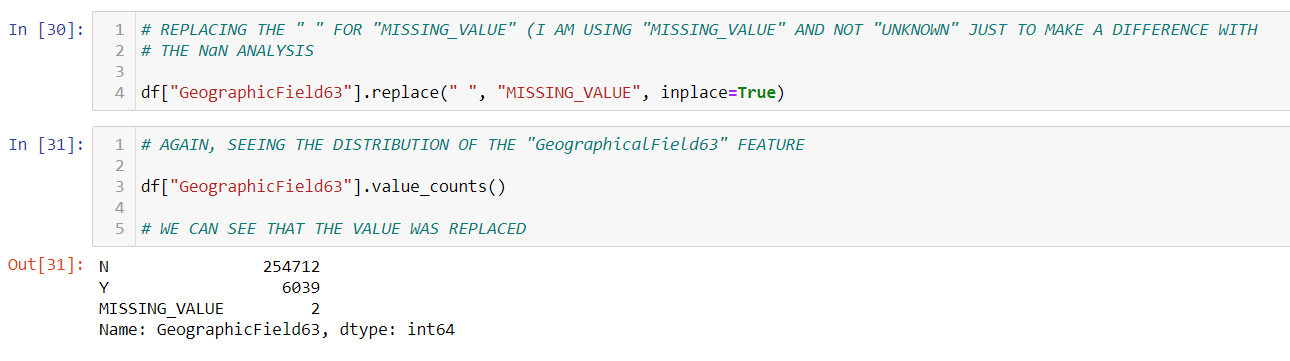


Type Check and Conversion

When doing the type check I realized there was one feature a blank value.

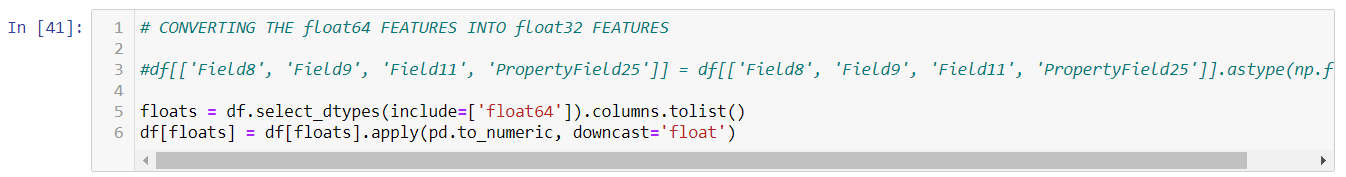


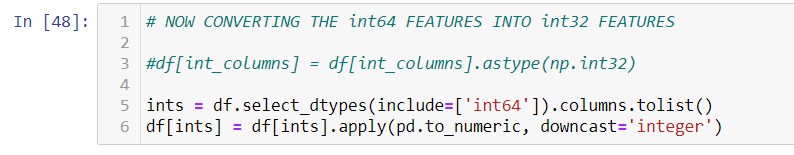
So, I decided to fill this value with the value “MISSING\_VALUE”.



This precious finding interrupted my Type Check and Conversion process but now that it is solved, I could actually do my Type Check and Conversion process.

In order to downcast the floats and integers I ran the following code:



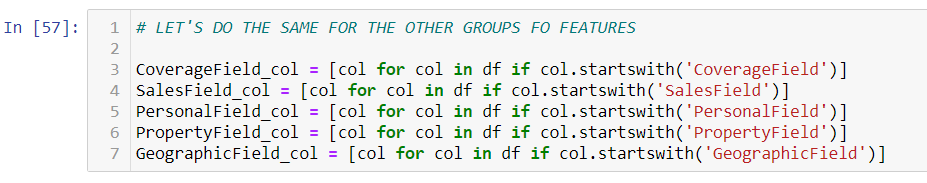


This code will downcast the feature to the lowest possible according to the values stored in the features.

Correlation Analysis and Dropping Features

I ran a correlation analysis to find the features that are highly correlated so I could drop them and keep only one of them. There is no point of keeping more than one feature that behaves in the same manner to others, that even makes the model worse.

I ran correlation among grouping by the following categories:

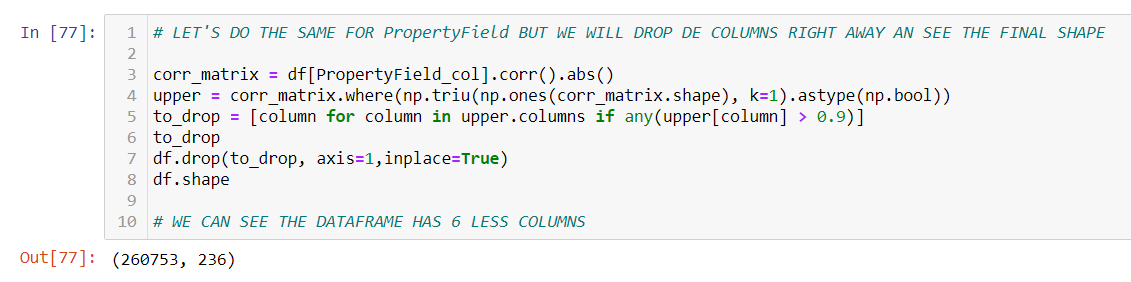


The I built correlation charts with seaborn to fin the highly correlated features. Here an example of only one of them.



In the previous image we can clearly see the correlation of the feature of the group CoverageField.

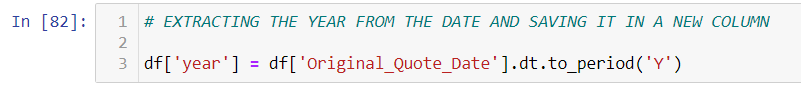
To drop the correlated features, I used the following code for each group of features:

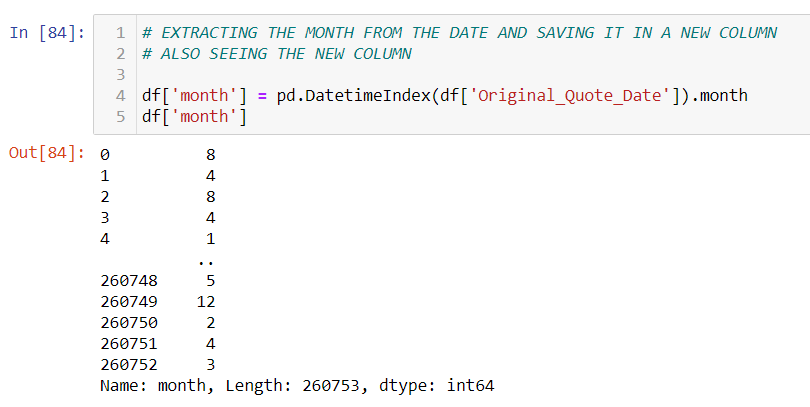


Feature Engineering

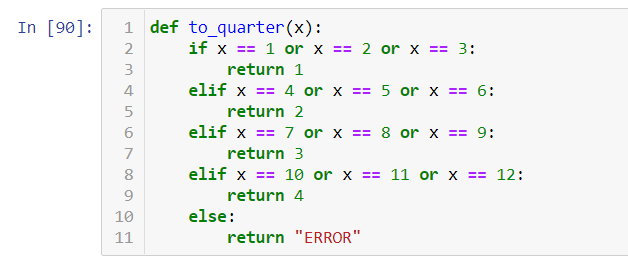
Since al the features are meaningless features because they do not have names, it was not possible to generate new feature or perform complex feature engineering. However, I used one of the 3 meaningful features Original\_Quote\_Date to generate features such as “year” and “month”.

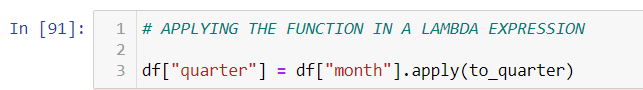
For that I used the following codes:

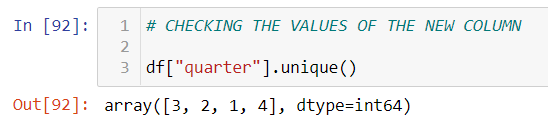




I also used a lambda expression to generate the feature “quarter”, however, later I realized that similar to year and month, there is a method to generate the “quarter” feature automatically. Since it was already too late and it was also good to practice lambda expressions, I used that to generate the “quarter” feature.



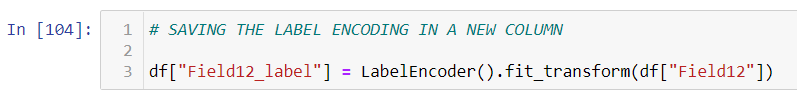




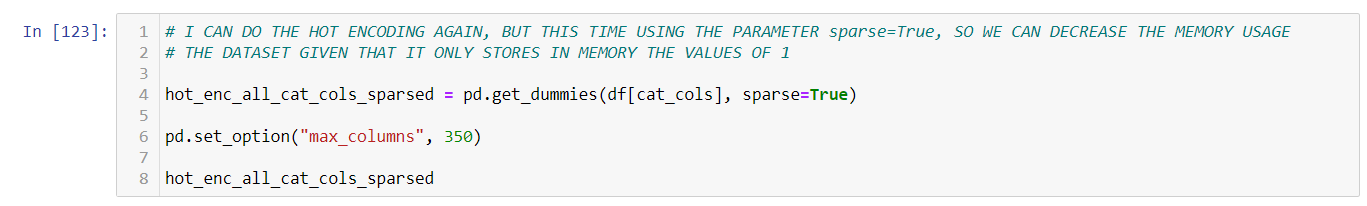
Process Categorical in Depth

For this part of the analysis I label encoded the feature Field12 just for practice purposes and I One Hot Encoded the rest of categorical features, given that I think they are unordered categorical features. Again, not knowing the meaning of the features makes it hard to determine what kind of categorical feature they are.

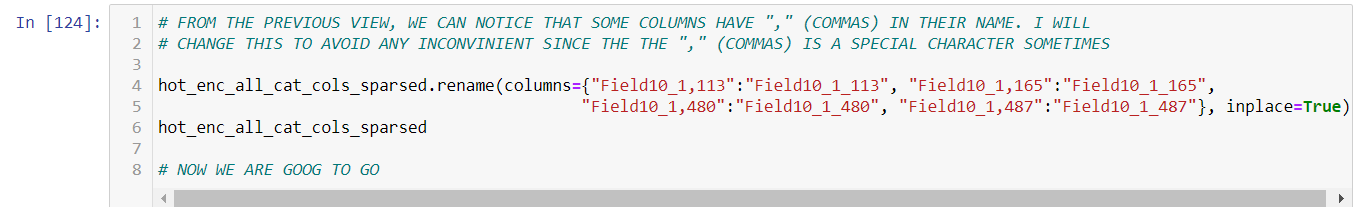
I Used the following code for the label encoder for the feature Field12.



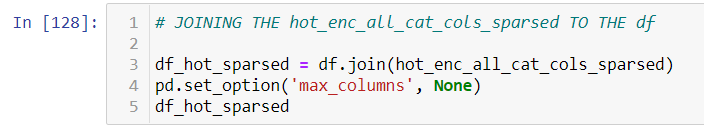
For the rest of categorical features thar were One Hot Encoded, the code was the get\_dummies which come integrated in pandas.



After doing the One Hot Encoding I realized one of the features had number feature in a text format and that the values were using comma to divide the thousands. So, to avoid having commas in the feature names I replaced the name of these generated One Hot Encoded features with the following code:



Since the Label Encoder and One Hot Encoder generate new data frames I had to join the data frames with a code like this one:

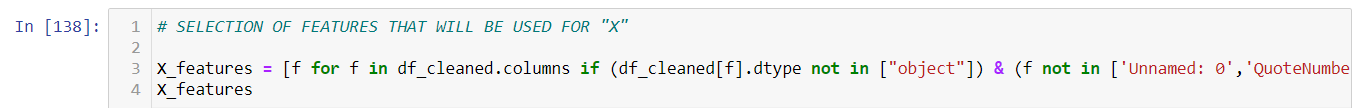


Splitting Data

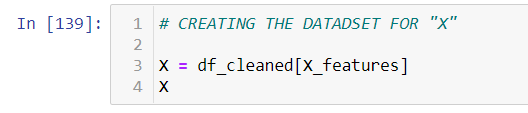
Now it was time to split the data into X (features) and y (target) and at the same time in train set and test set.

To separate the data set in features and target (X and y) I ran the following code:

This code is to just select the features part of the dataframe. I am excluding the objects since these were already converted into numbers with the help of the Label Encoder and the One Hot Encoder. I am also excluding some specific features that would noy help such as the QuoteNumber.

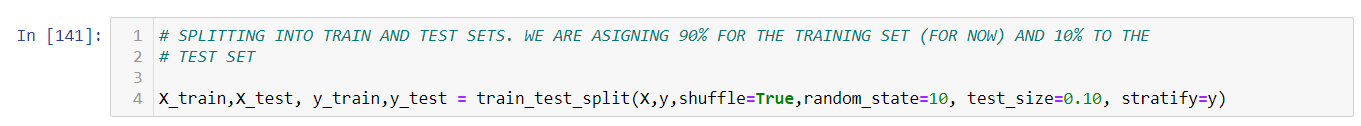


Then I create X and y like this:





Now to split in Train and Test I would run this code:



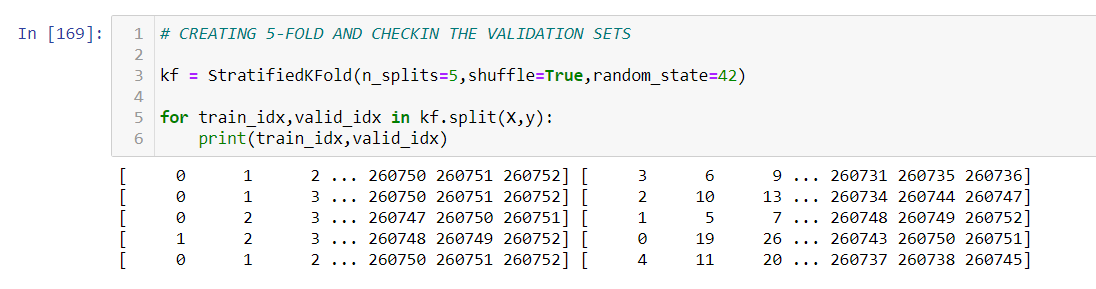
As we can see, the set was divided in 90% train set and 10% test set and within the test set it will be performed a 5-Fold Cross Validation creating validation sets within the train set.

LightGBM Model

Now that the split was done, I tested model like Decision Tree and Random Forest, but the LightGBM turned out to be the best one for my predictions, so I will use this model.

In the LightGBM model I included a 5-Fold cross validation to avoid overfitting. This 5-Fold validation will alternate using a different validation set from the training set 5 times and train the model to avoid overfitting, bias, or variance. It will be exactly be like training the model 5 times with different test set that within the 5-Fold Cross Validation will be named validation sets. The real test set will be used at the end.

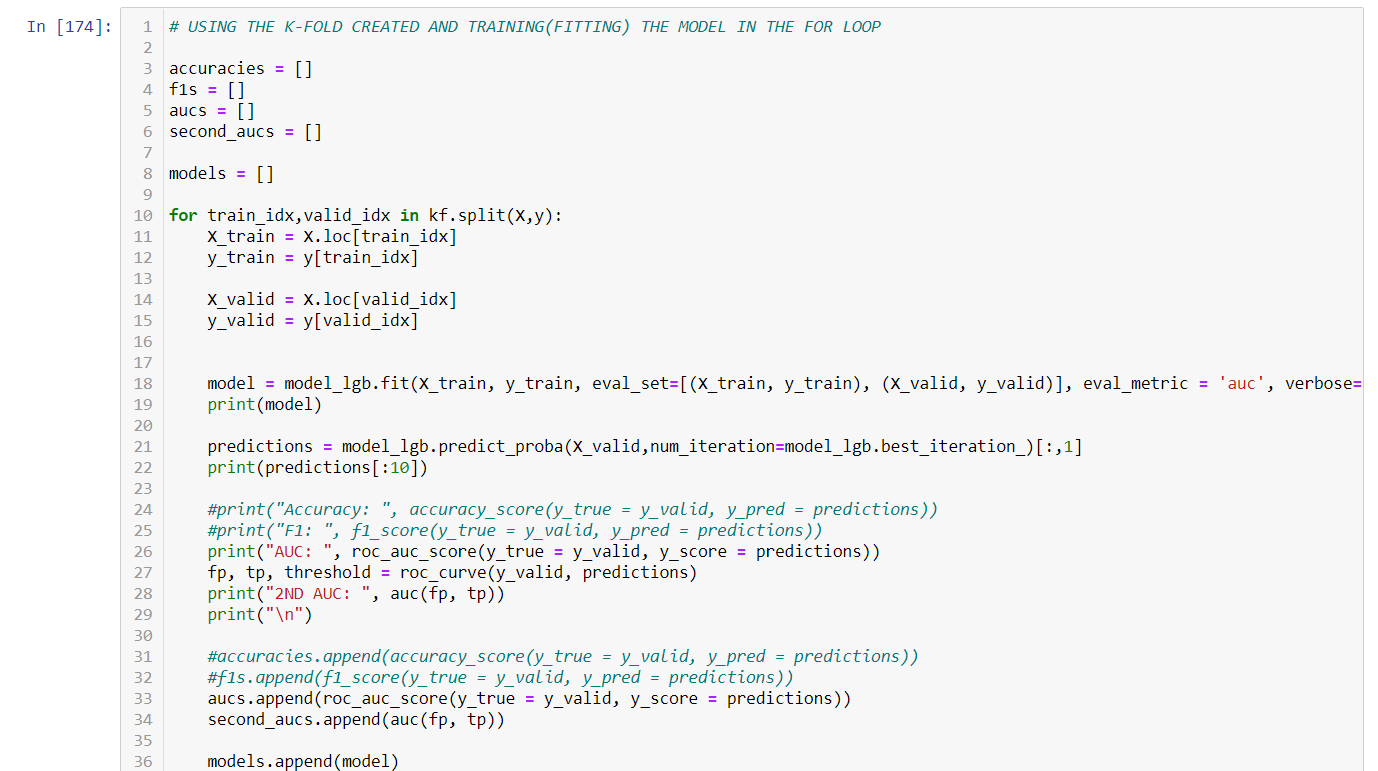
This code will create the 5-Fold Cross Validation:



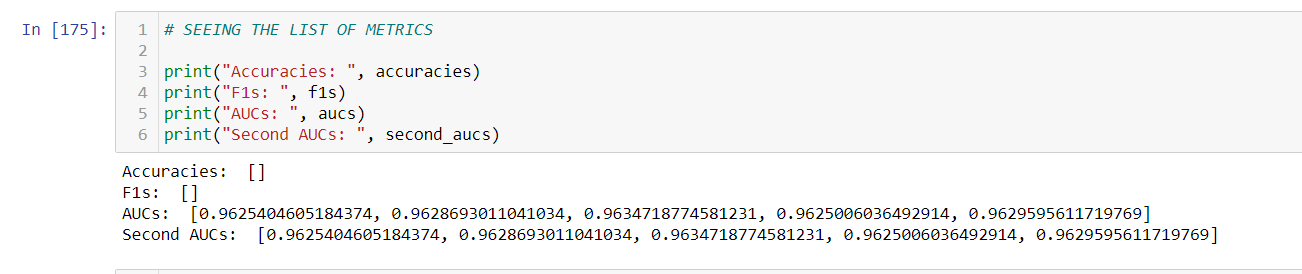
The LightGBM Model Parameters are the following:



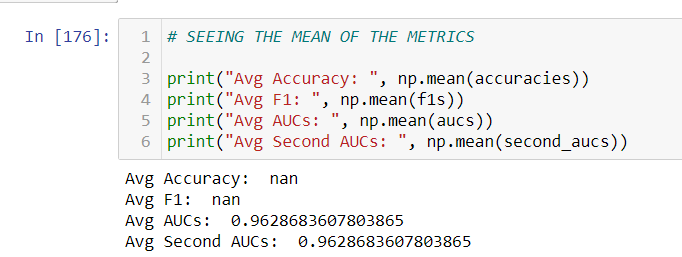
To train the model using the 5-Fold Cross Validation we created we will use this code:



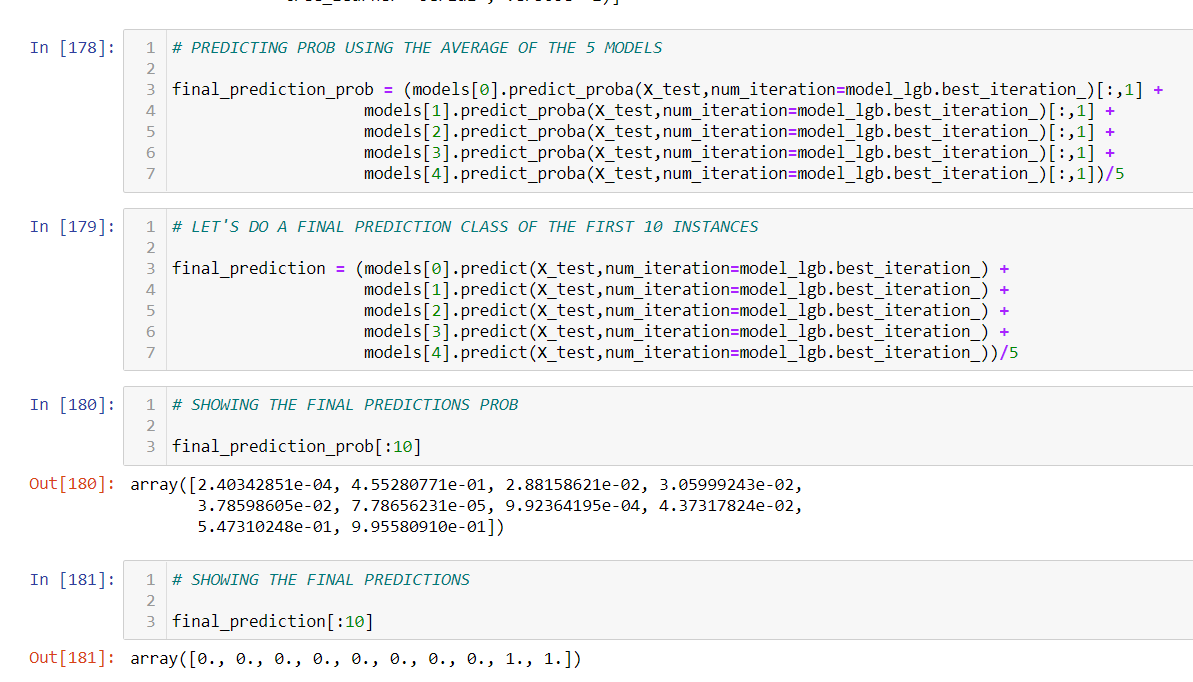
Finally, I will show the results of the model. First, we will see the result of each of the 5 models that were run in the 5-Fold Cross Validation:



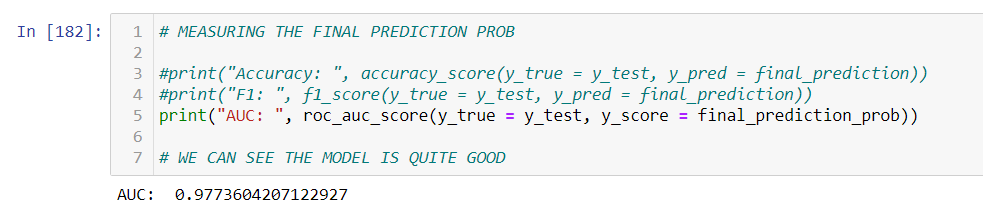
And now the average of the results:



And the predictions both as an absolute value and also as a probability of the customer buying the insurance are the following.



Finally, the AUC result when using the test set that was never seen by the model:



Conclusion

Right after training the model we saw that the AUC of the model was 0.962868. This is a more than a good AUC. And moreover, when testing the model with unseen data (the test set), the result of the AUC was even higher: 0.977360. This means the model worked even better on unseen data.

We can definitely conclude that, even though it was hard to deal with features that do not have a name or a meaning, the AUC result was very good and that this model is definitely generating excellent predictions.

Comments

Along with this report, the Jupyter Notebook will be included and also a group of .py files, since the code developed in the Jupyter Notebook was also transferred in pycharm to have a clean code organized by different .py files.

In other words, the Jupyter Notebook was used as draft or notebook for the code and the .py file is the clean code.

The CSV file can be found as well under the name of “train.csv”.

Bibliography

[www.kaggle.com](http://www.kaggle.com)

[www.scikit-learn.org](http://www.scikit-learn.org)

[www.pandas.pydata.org](http://www.pandas.pydata.org)

[www.numpy.org](http://www.numpy.org)

[www.matplotlib.org](http://www.matplotlib.org)

[www.seabor.pydata.org](http://www.seabor.pydata.org)

<https://medium.com/bigdatarepublic/advanced-pandas-optimize-speed-and-memory-a654b53be6c2>

<https://christophm.github.io/interpretable-ml-book/interaction.html>