Abstract

We aim to use different classification techniques to predict if customers will subscribe to a term deposit based on client and campaign data

Data mining for business intelligence

Supervised Learning Project Report

Done by:

Eliza Redwine, Deepak Gaur, Karishma Bhatnagar, Priya Thomas, Kinjal Shah, Vinay Temburnikar

Contents

[Predicting if customers will subscribe to a term deposit based on client and campaign data 2](#_Toc5024982)

[Introduction to the data set 2](#_Toc5024983)

[Data Preparation 2](#_Toc5024984)

[Exploratory data analysis of the chosen data set 2](#_Toc5024985)

[Reading the data 2](#_Toc5024986)

[Exploratory data analysis 3](#_Toc5024987)

[Missing value check 3](#_Toc5024988)

[Visualizing the data 3](#_Toc5024989)

[Feature Engineering 4](#_Toc5024990)

[Split data set into training and test data set 5](#_Toc5024991)

[Splitting labels from train and test data 6](#_Toc5024992)

[Model Specification 6](#_Toc5024993)

[Decision Tree Classifier 6](#_Toc5024994)

[Linear Discriminant Analysis Classifier 9](#_Toc5024995)

[Random Forest Classifier 10](#_Toc5024996)

[Logistic Regression 13](#_Toc5024997)

# Predicting if customers will subscribe to a term deposit based on client and campaign data

## Introduction to the data set

We are using a data.world data set (<https://data.world/data-society/bank-marketing-data>), this data set related with direct marketing campaigns (phone calls) of a Portuguese banking institution. It explores the outcome of customer’s decision to subscribe to a term deposit based on the known data of the client and last contact made in the current marketing campaign. We are only using the bank\_full.csv file for the purposes of this project.

## Data Preparation

### Exploratory data analysis of the chosen data set

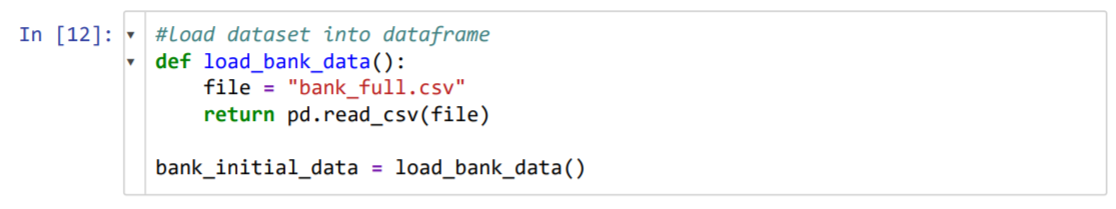
In total, there are 17 columns and 45211 records in this data set.

|  |  |  |
| --- | --- | --- |
| **Column** | **Data type** | **Representation** |
| Age | Integer | Client’s age |
| Job | String | Type of job the client holds |
| Marital | String | Client’ marital status |
| Education | String | Client’s highest level of education |
| Default | Boolean | Does client have credit in default |
| Balance | Integer | Client’s average yearly balance |
| Housing | Boolean | Does client have a housing loan with the bank |
| Loan | Boolean | Does client have a personal loan with the bank |
| Contact | String | Contact communication type |
| Day | Integer | Day last contact made |
| Month | String | Month last contact made |
| Duration | Integer | Duration of last contact in seconds |
| Campaign | Integer | Number of contacts performed during this campaign and for this client (includes last contact) |
| Pdays | Integer | Number of days that passed by after the client was last contacted from a previous campaign |
| Previous | Integer | Number of contacts performed before this campaign and for this client |
| Poutcome | String | Outcome of the previous marketing campaign |
| Y | Boolean | Has the client subscribed a term deposit? |

We will use multiple models to try and predict the variable Y.

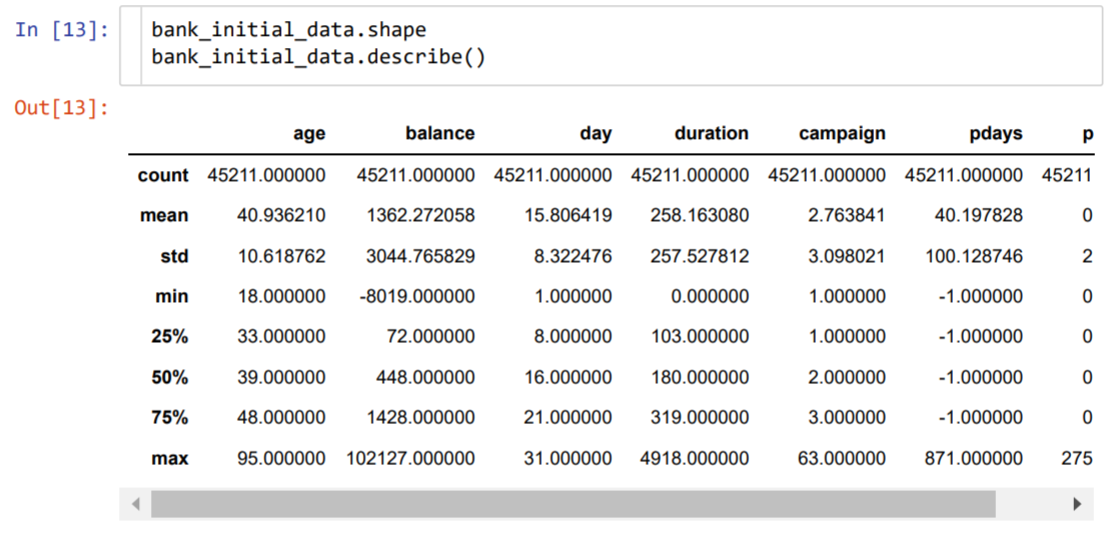
### Reading the data

Before we apply any of the models, we first read the csv file and store it in a data frame.



### Exploratory data analysis

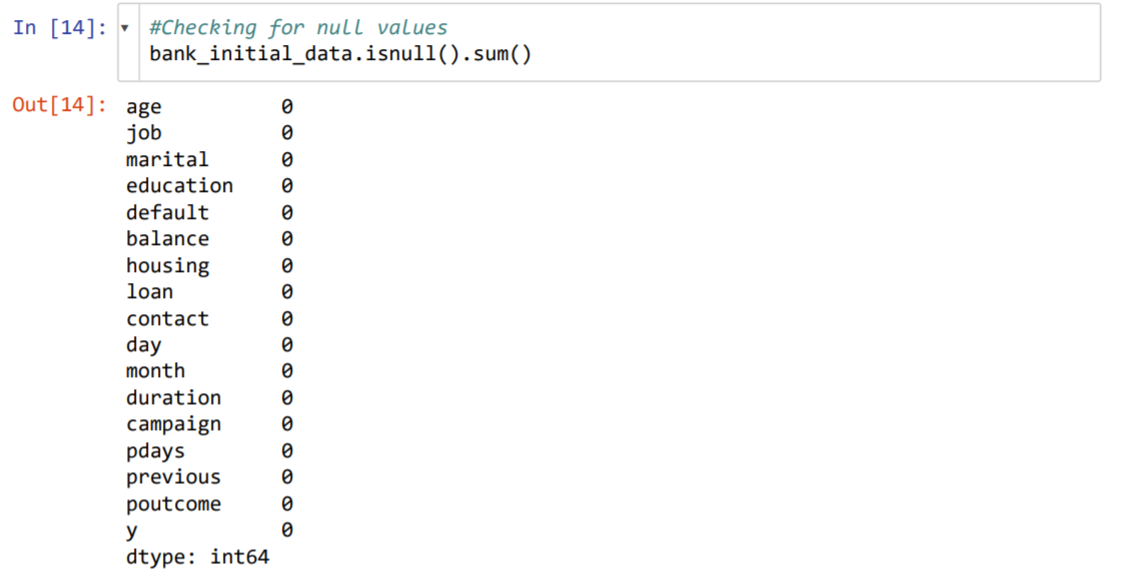
We view the data to have a better understanding of our data set.



Dataset looks clean and tidy from the above statistics, min, max and mean values for all the numerical data seems logical and correct.

### Missing value check

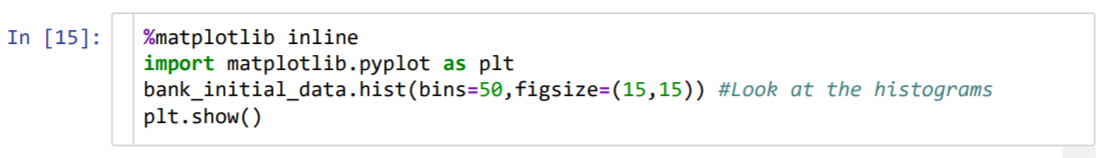
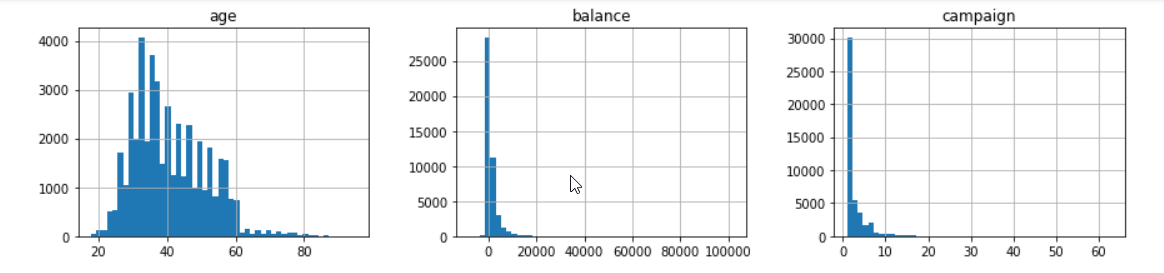
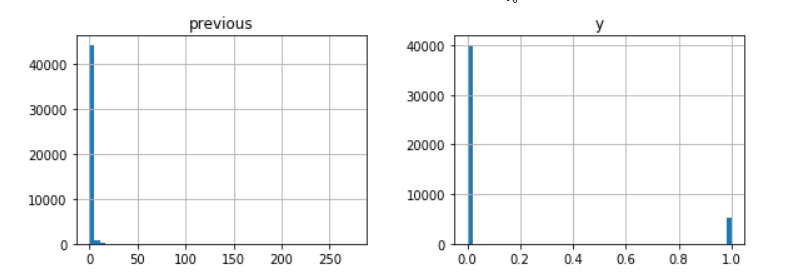
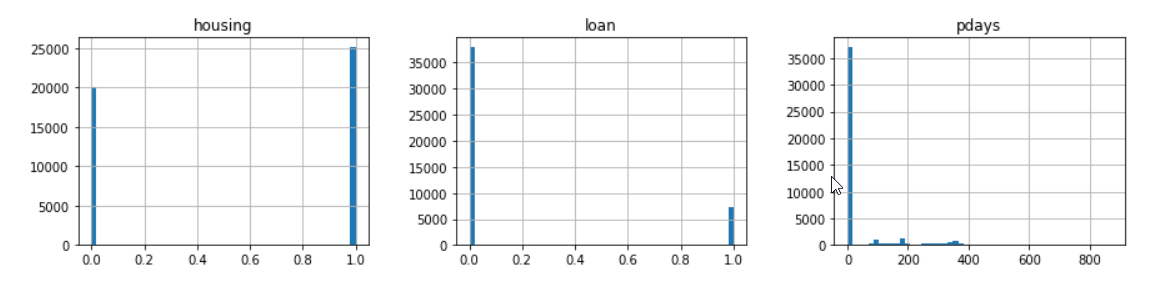
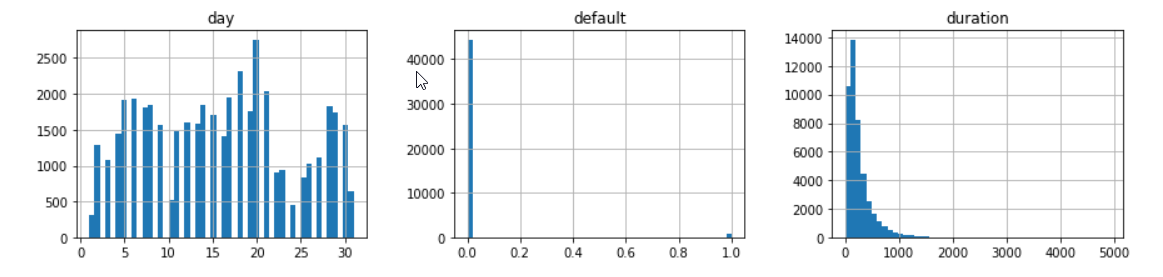
We check whether any null or missing values are present in our current data set before proceeding further with our analysis and applying our models to the data.



As we can see from the results, we have 0 null values in the data set.

### Visualizing the data

We plot histograms for all the numeric integers to see their distribution.

****

## Feature Engineering

We first drop the variable, Duration as it has a significant impact on the target variable, Y. If duration is 0, then Y will be No. This variable will be discarded as we intend to have a realistic predictive model.

We convert all our non-numeric features to numeric because sklearn’s fit function only takes numeric variables.

We convert the following 4 Boolean variables into numeric variables (True=1, False=0). Variables are:

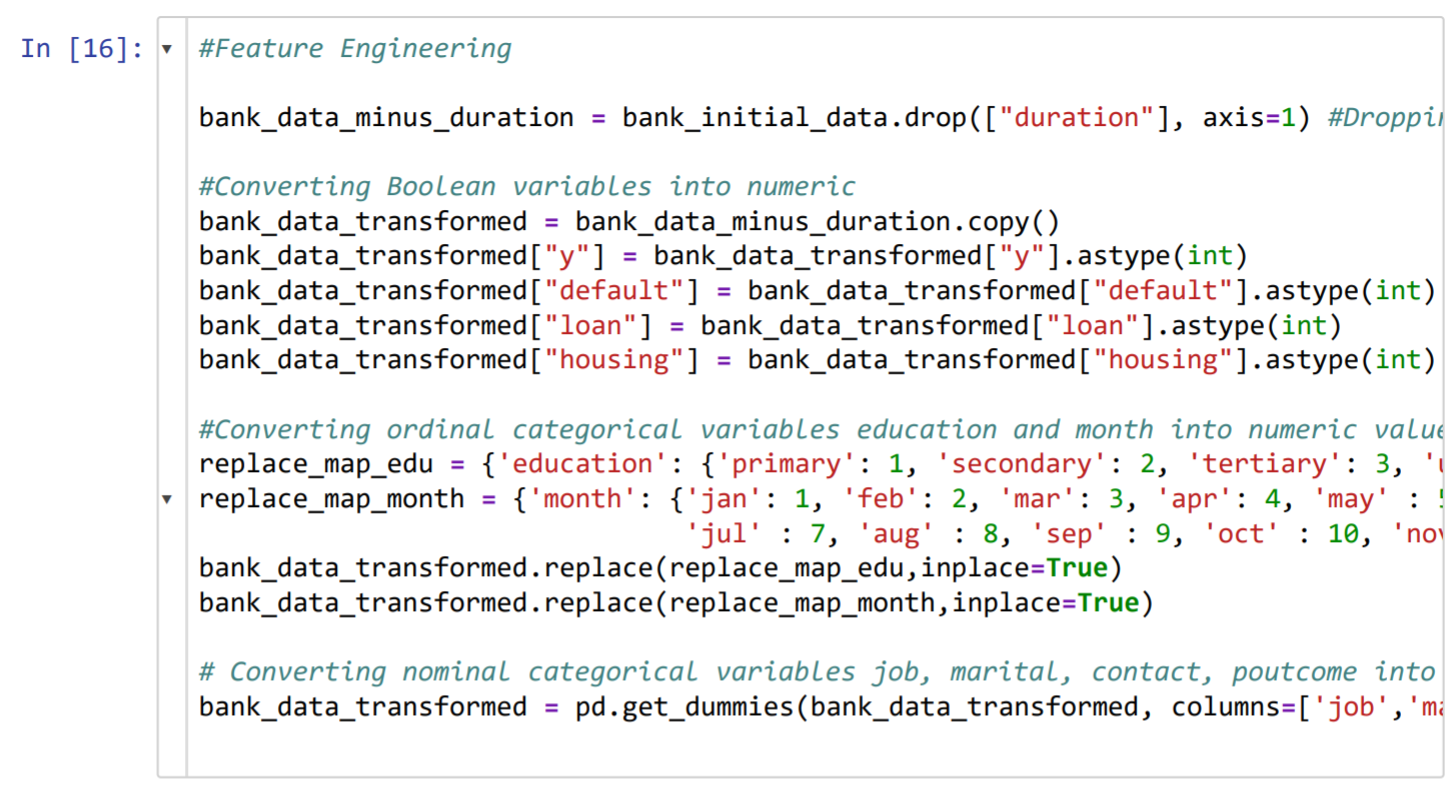
* Y
* Default
* Loan
* Housing

Next, we convert the 2 ordinal categorical variables into numeric variables by manually assigning numbers to each factor. Variables are:

* Education
* Month

Finally, we convert the nominal categorial variables into numeric variables using one-hot-encoder: Variables are:

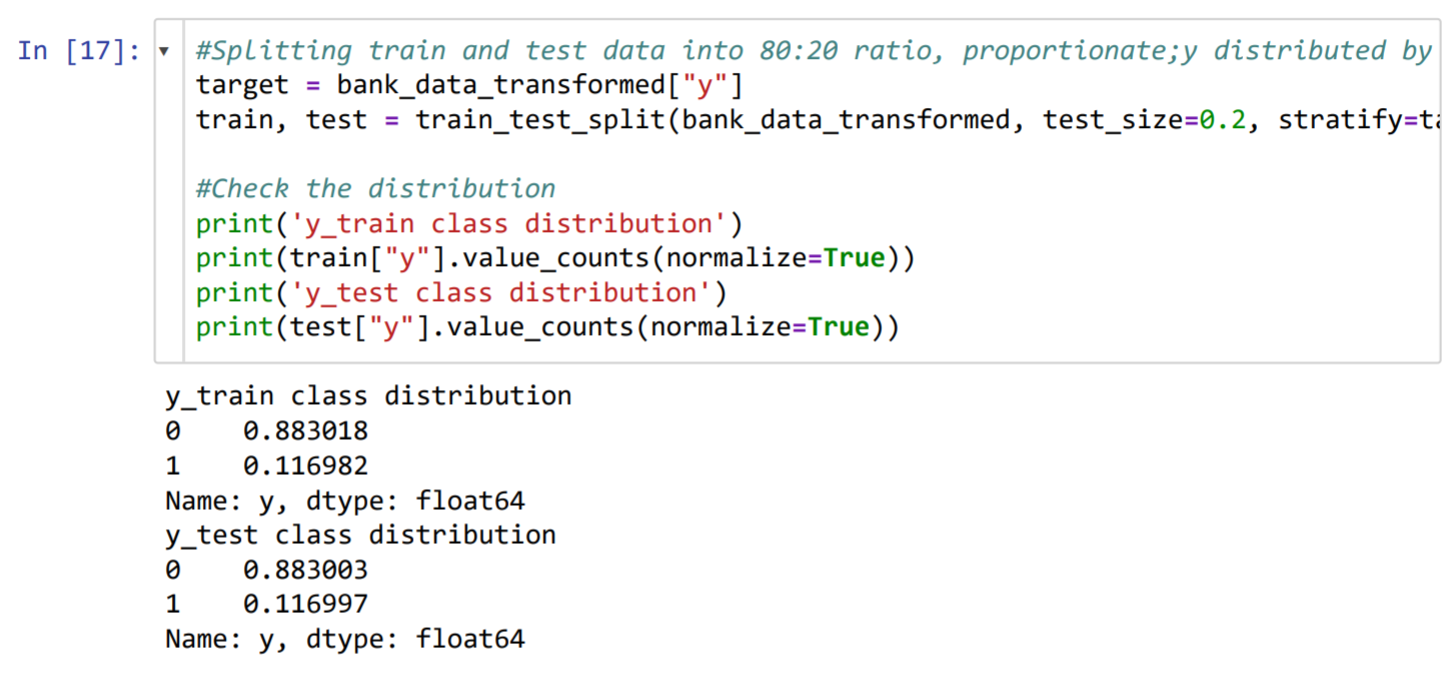
* Job
* Marital
* Contact
* Poutcome



### Split data set into training and test data set

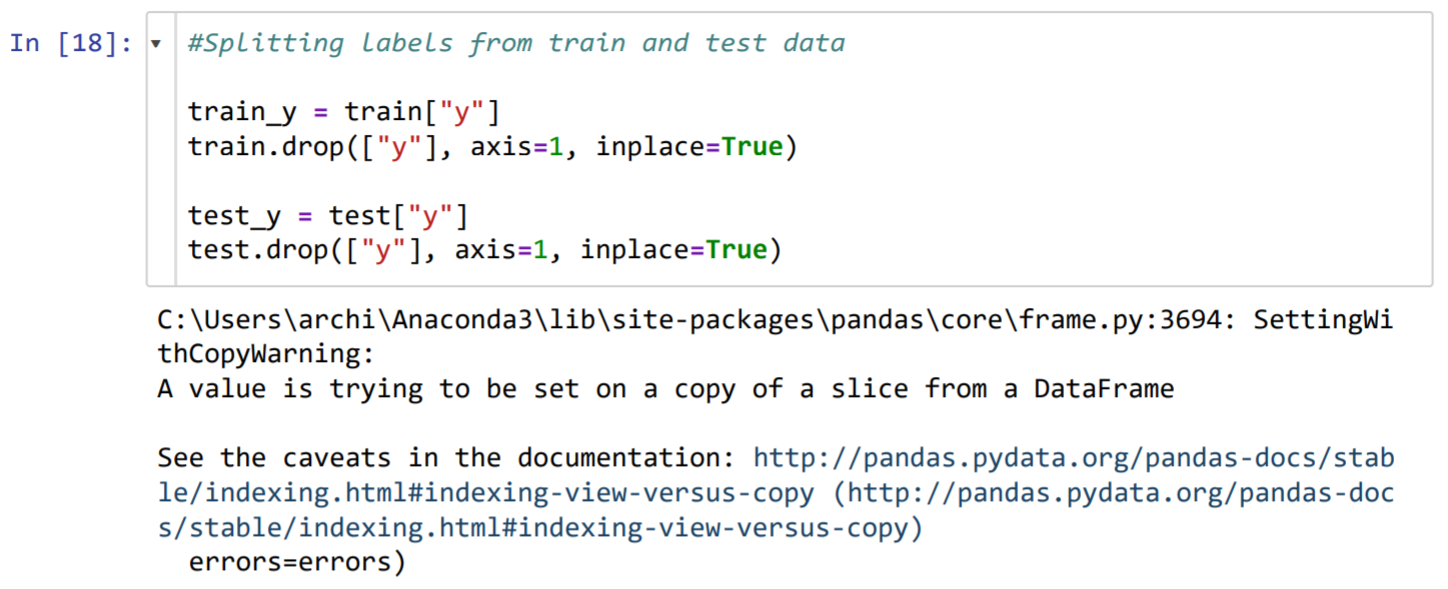
We split the dataset into 80:20 (train: test) data ratio proportionately. The training set, train, will be used to train the model. We have the test dataset, test, in order to test how well model generalizes on the unknown data.

We are using stratified splitting on the dependent variable y to make sure that both train and test dataset have equal proportion of both the classes.



### Splitting labels from train and test data

We split the target variable from both out train and test data.

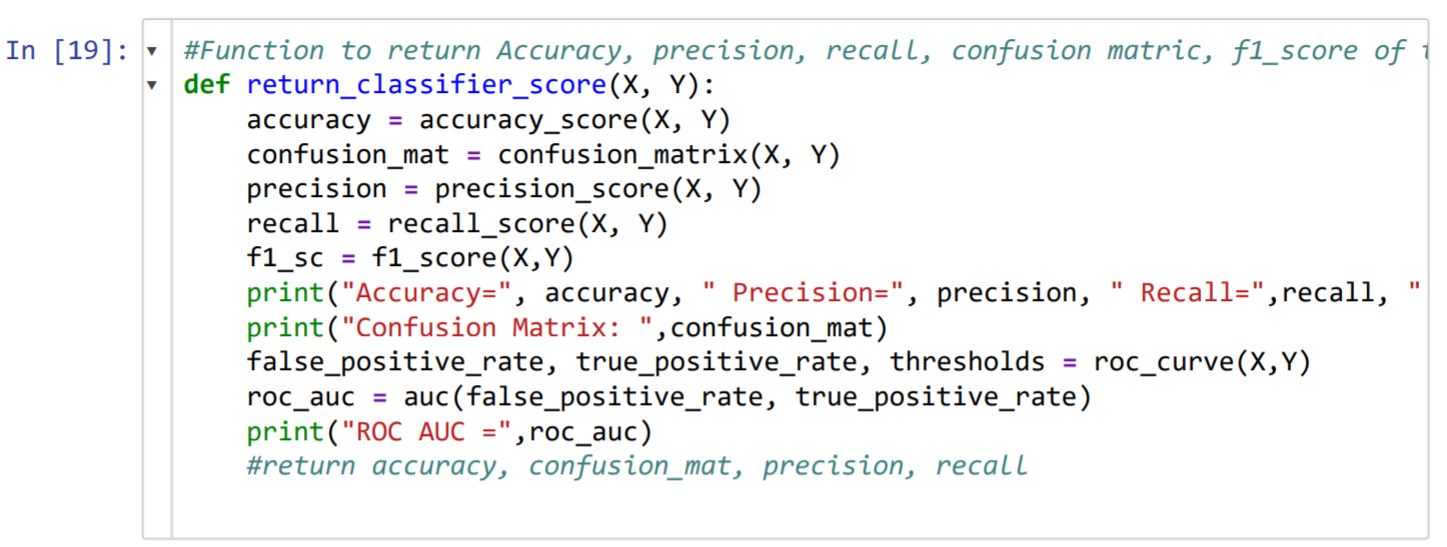


## Model Specification

### Decision Tree Classifier

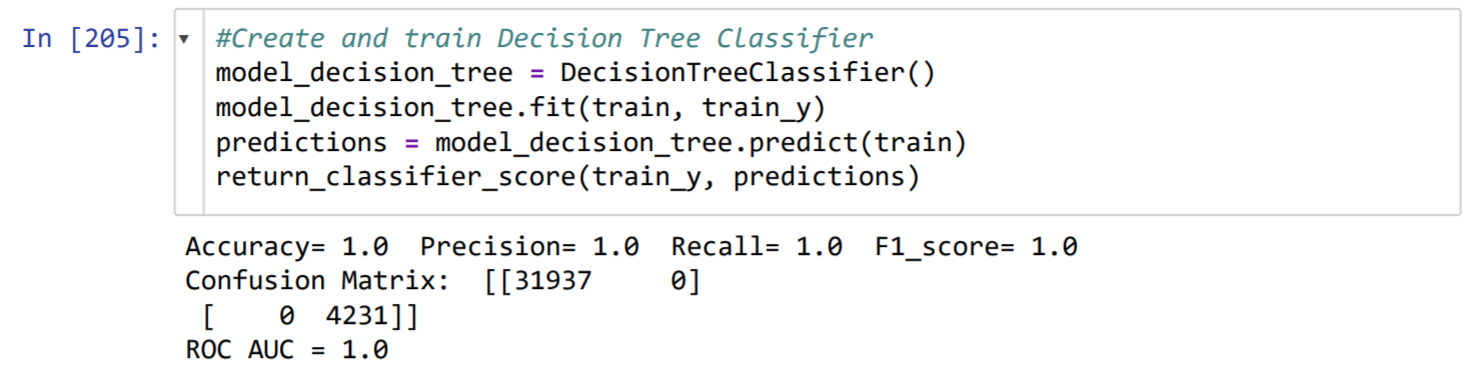
The first classifier we will use on our data set is the Decision Tree model. We define a function, return\_classifier\_score, to return the following scores:

* Accuracy
* Precision
* Recall
* Confusion matrix
* F1 \_score



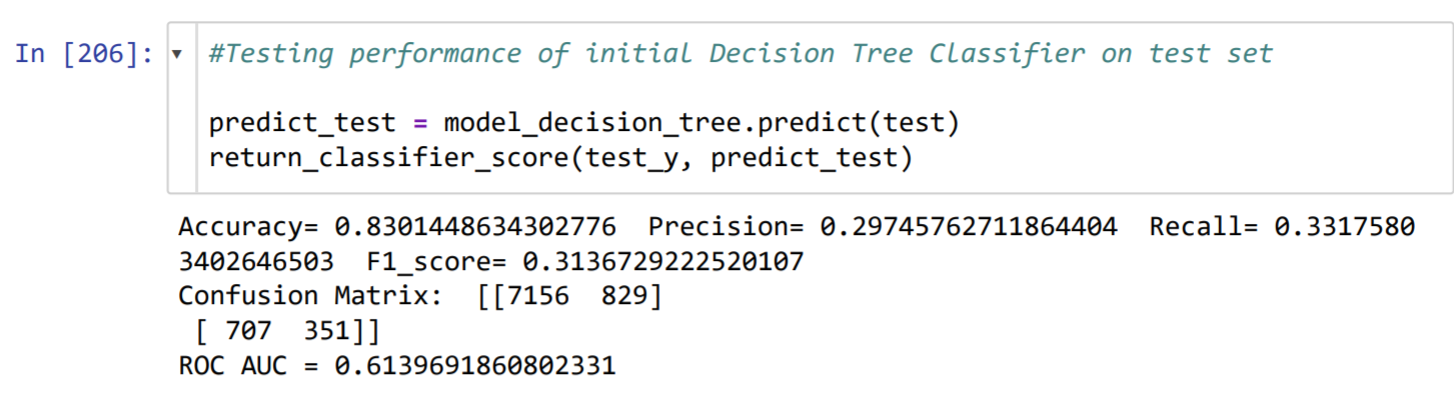
#### Create and train decision tree classifier

We create and train the initial Decision Tree Classifier with our train data.



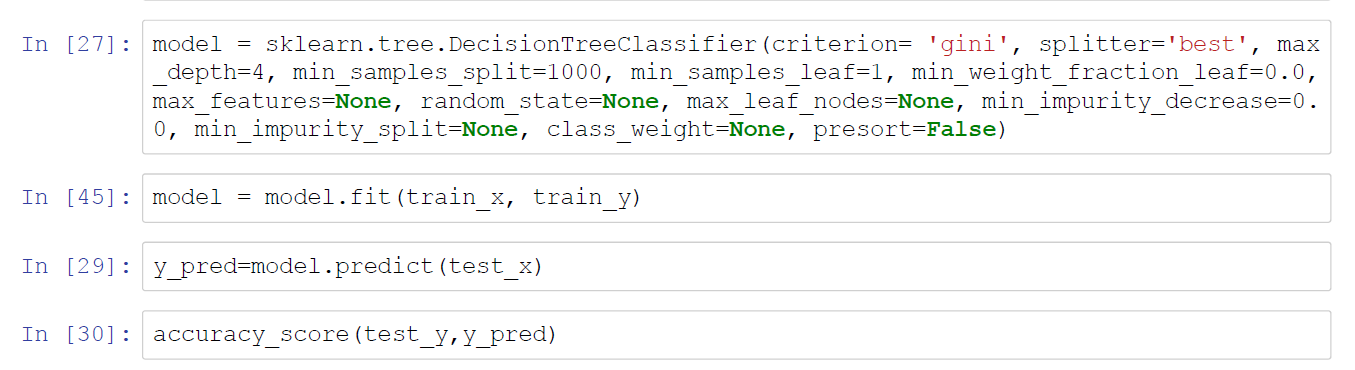
#### Test performance of initial decision tree classifier on test data set

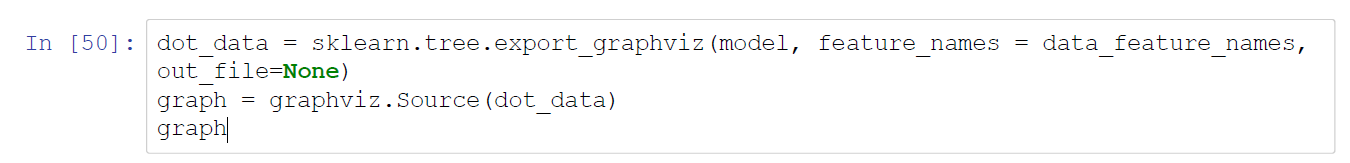
With the initial Decision Tree Classifier created, we test it on the test data set.

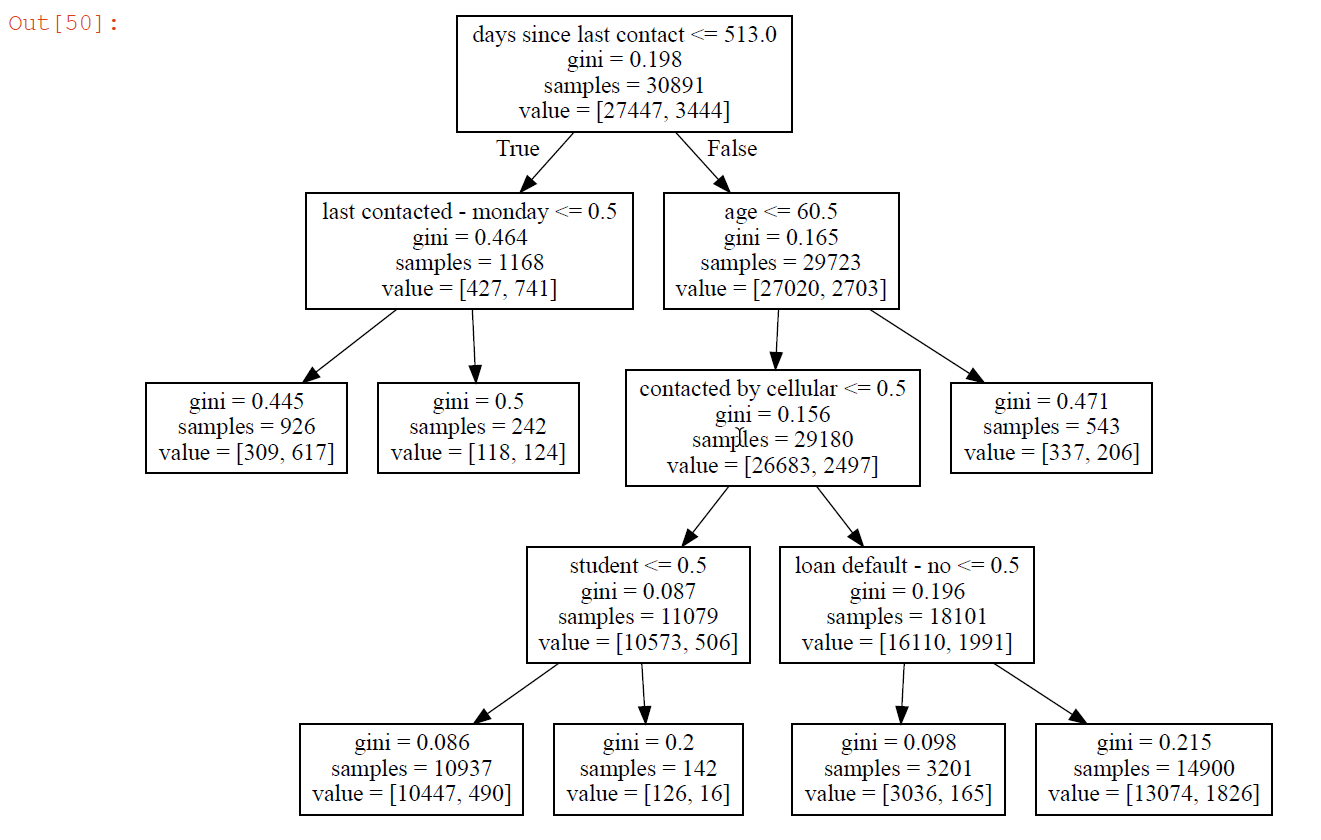


From the results we see that there is scope for the decision tree classifier to be more precise. In the next few steps, we tweak the model to be better.

#### Graphical display of the decision tree model



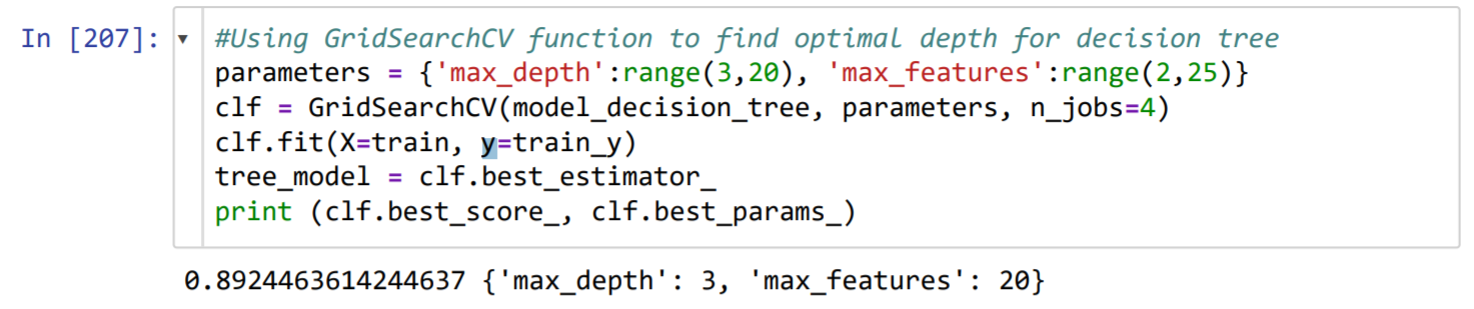




The above decision tree helps us understand the logic that model is using when actually predicting the class of the given data points.

#### Find optimal decision tree depth

We use the GridSearchCV function to find the optimal depth and features for the decision tree.



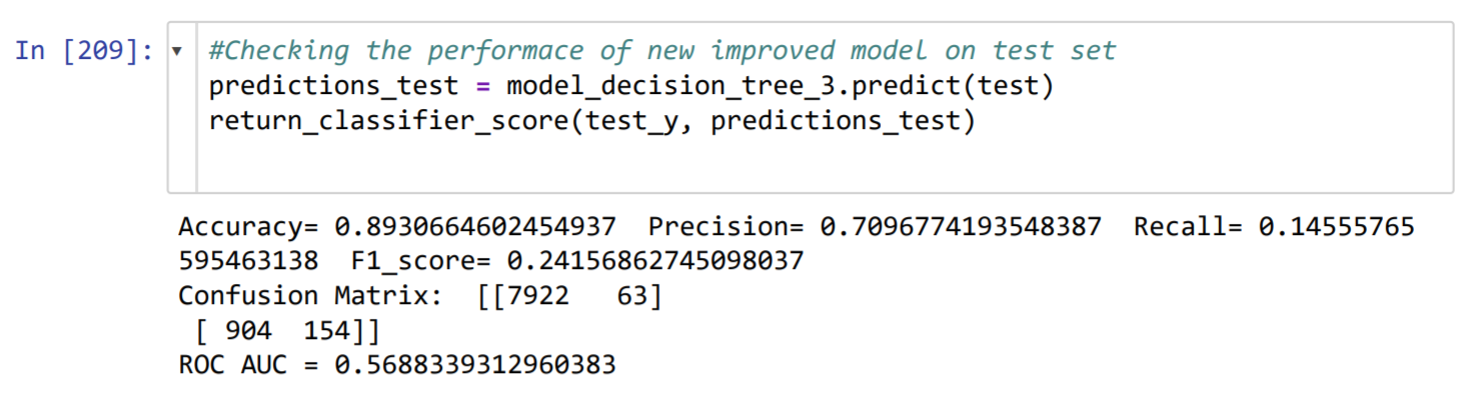
#### Creating new decision tree model based on the above suggestion

We create a new model with the output provided by the GridSearchCV function.



#### Test performance of improved model on test data set

We check the performance of the new improved model on the test data set.



As we can see from the results above, performance of the new model is much better than the initial decision tree model in terms of accuracy and precision. This model achieves 89% accuracy and 71% precision. However, the recall for the new model is low.

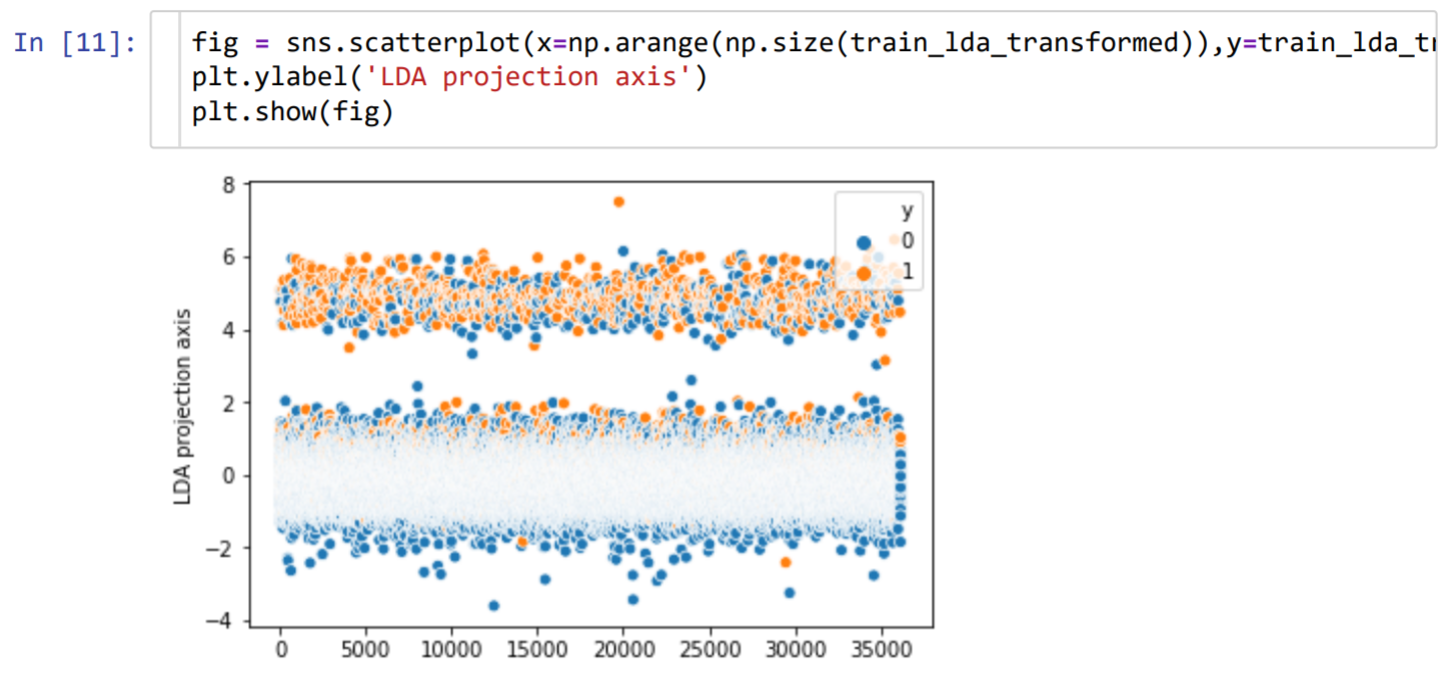
### Linear Discriminant Analysis Classifier

The second classifier we will use on our data set is the Linear Discriminant Analysis. This algorithm tries to reduce dimensionality of the data by projecting data points in the lower dimension such that it is easier to classify different classes.



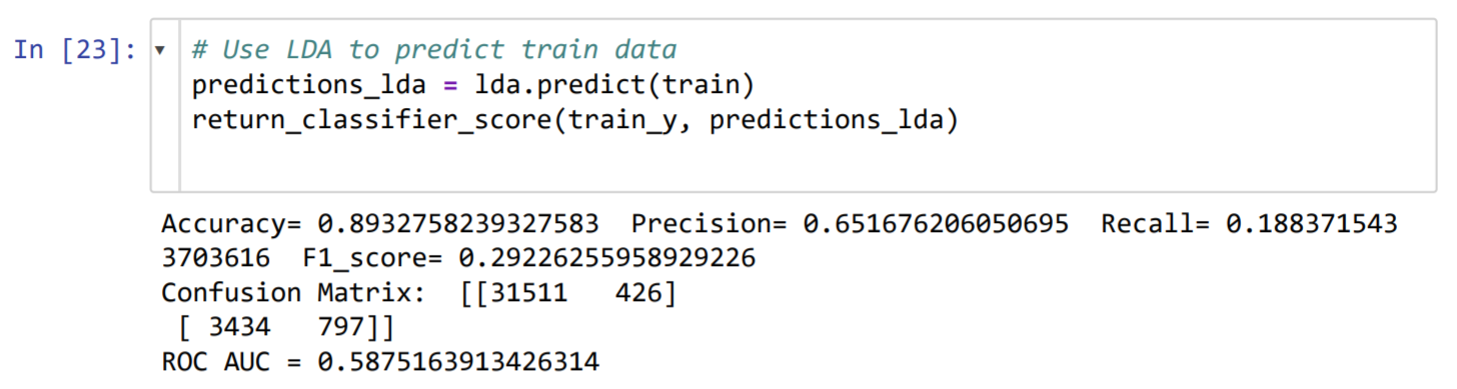
#### Create scatter plot to view transformed data

We create a scatter plot with our transformed data to visualize the class scatter.



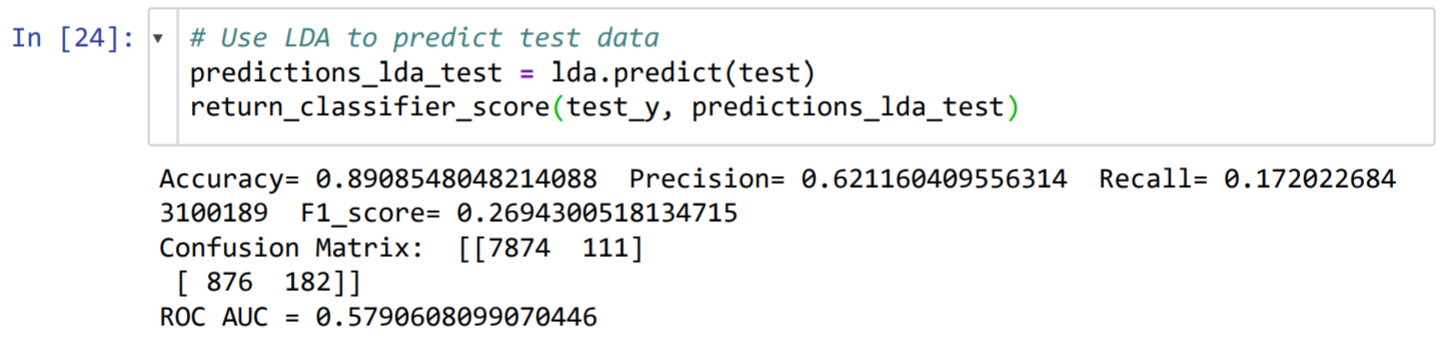
We can see that there is a good separation near the x-axis when y=0. However, both the classes are quite mingled up on the other part and they are not classified well.

#### Test performance of LDA on train data set

We test the performance of the LDA model on the train data set.

#### Test performance of LDA model on test data set

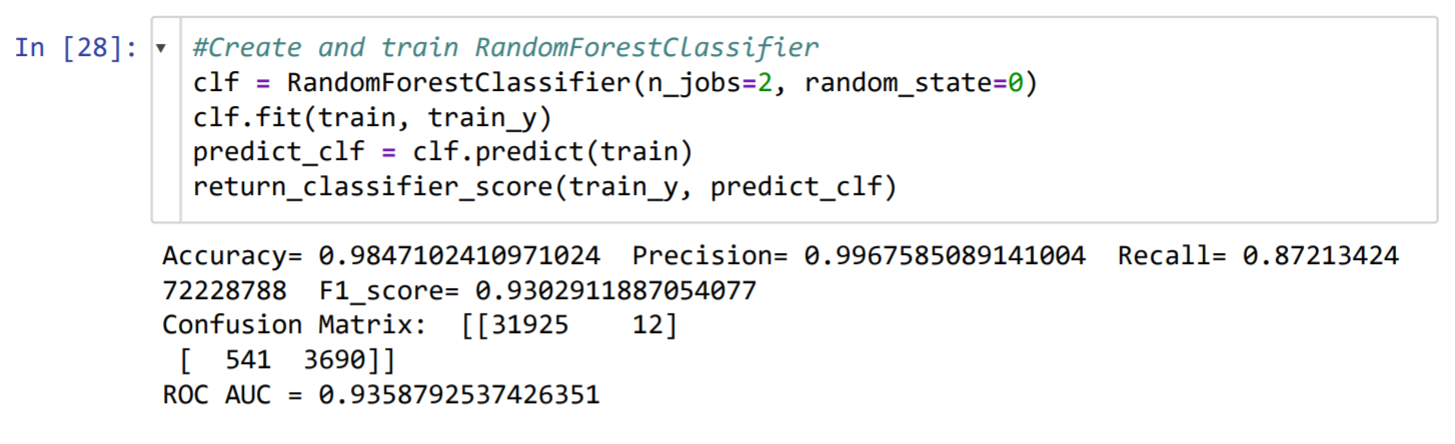
We then test the performance of the model on the test data set and view the results.



As we can see from the results above, this model achieves 89% accuracy, 62% precision and 17% recall. The model performance is relatively close to the decision tree model’s performance.

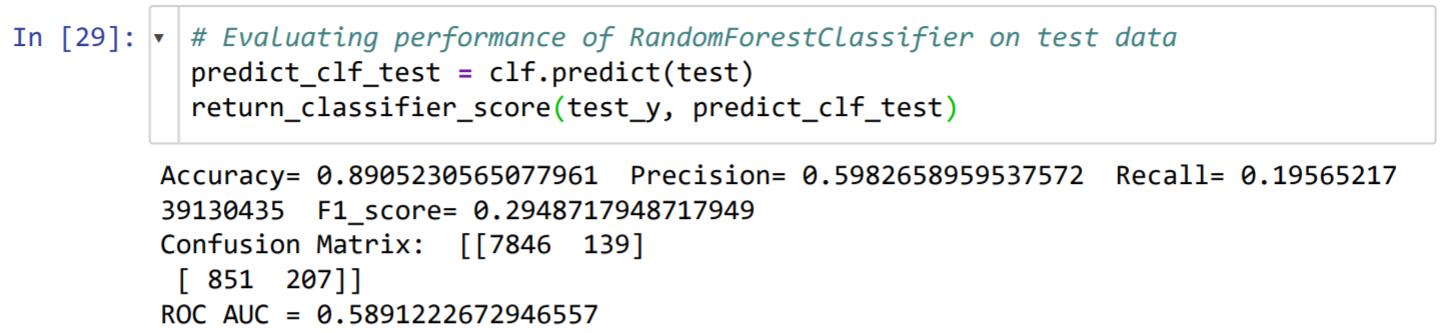
### Random Forest Classifier

The third classifier we will use on our data set is the Random Forest classifier.



We can see immediately that the accuracy and precision of the model is very high at 98% and 99% respectively.

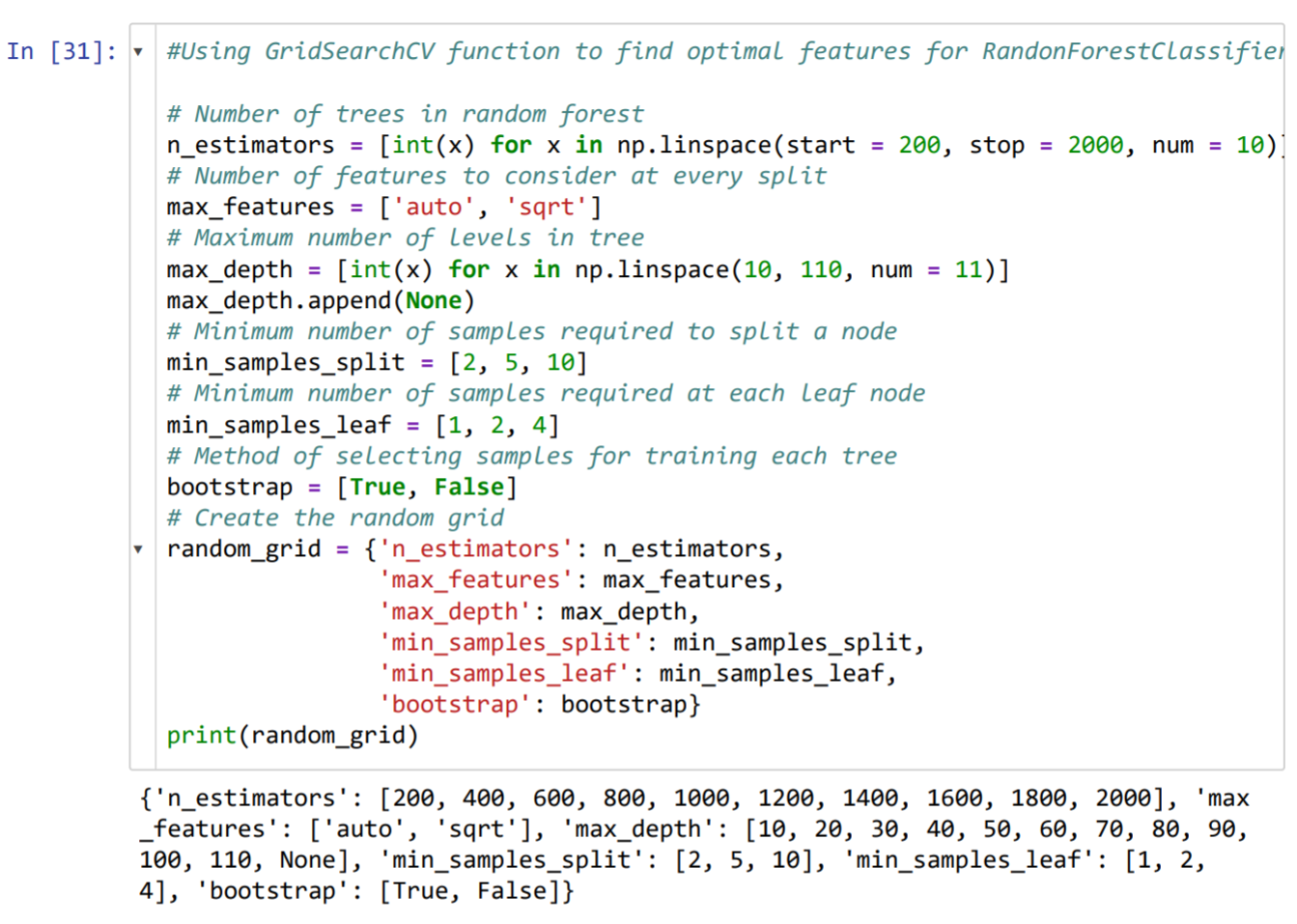
#### Test performance of model on test data set



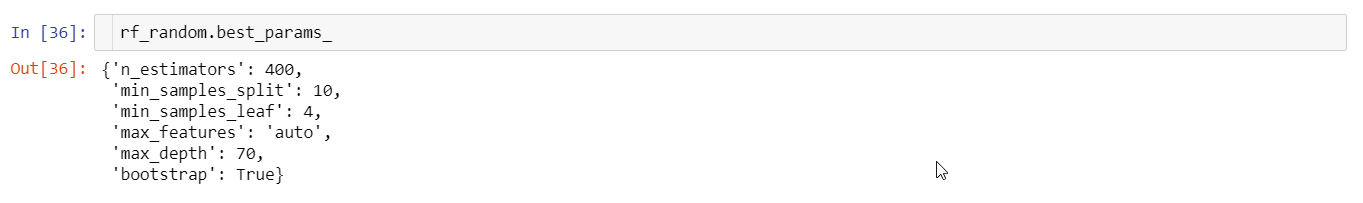
From the results, we can see that the accuracy and precision of the model reduces to 89% and 59% respectively. There is scope for the model to be more precise. In the next few steps, we tweak the model to be better.

#### Find optimal features for the model

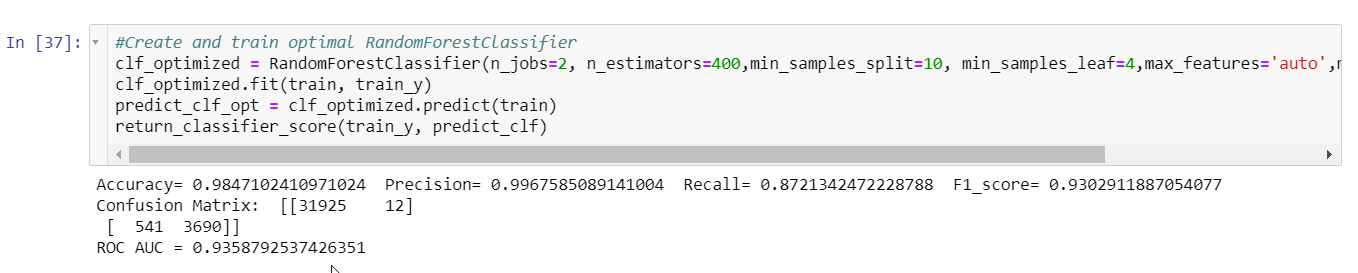
We use the GridSearchCV function to find the optimal features for the model.

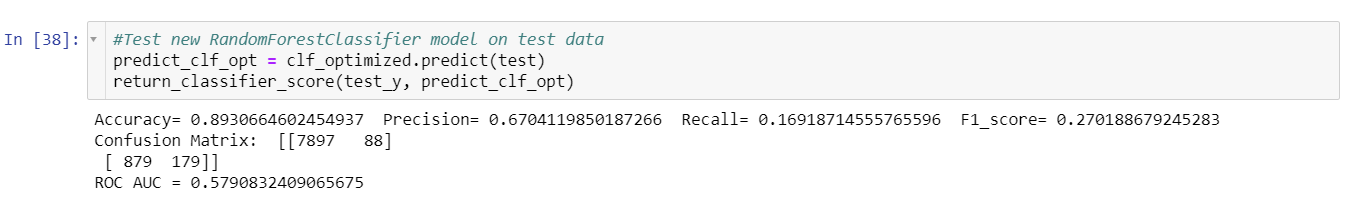


We are using GridSearchCV function to try to find the optimal hyper-parameters for Random Forest Classifier.



As per the parameter suggestions received from GridSearchCV function in the above figure, we create the new Random Forest Classifier model and test its performance on the test data.





As we can see, precision improved the earlier RFC model from 59% to 67%, however we lost a little bit on recall and f1 score.

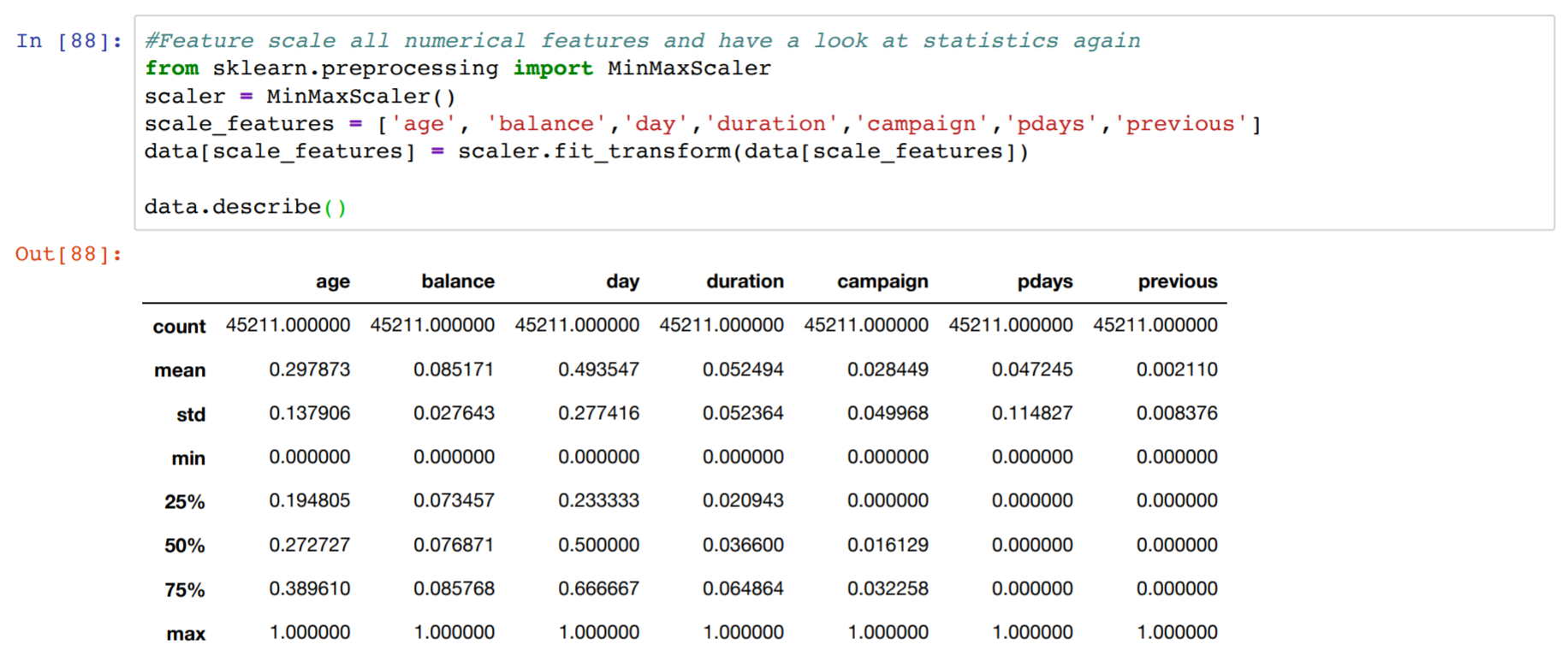
### Logistic Regression

The forth model we applied on our data set was Logistic Regression. Since we have already seen how the data looks like in the data exploration we conducted earlier, we can go straight into feature scaling of the numerical variables and converting data into independent and dependent values.

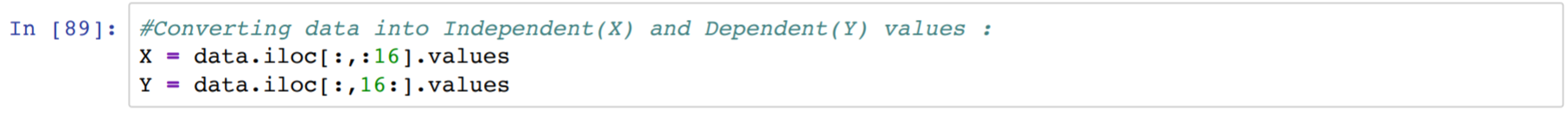
For logistic regression, we will start over with a new data set that has not yet had its categorical variables converted into numerical variables.

#### Feature scale numerical variables

We feature scale the numerical variables as there is high discrepancies in their ranges.

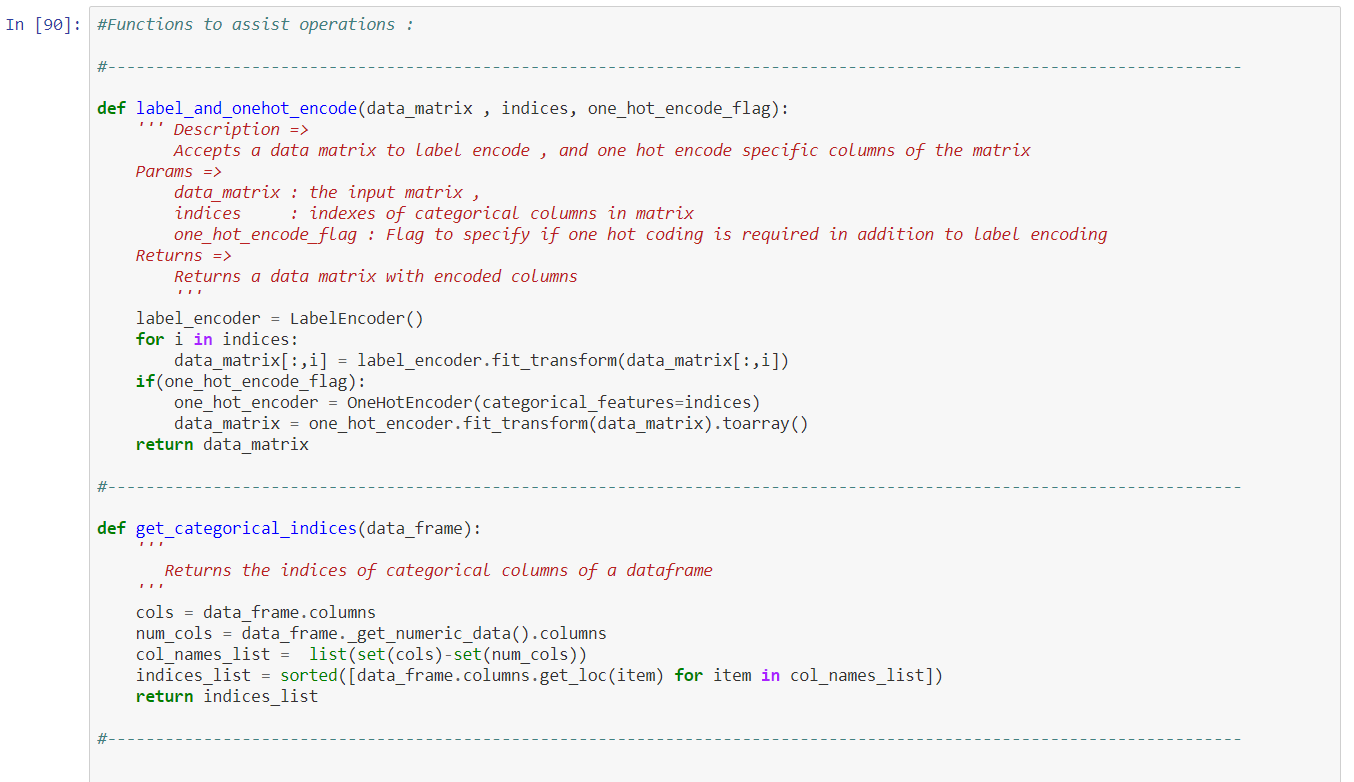


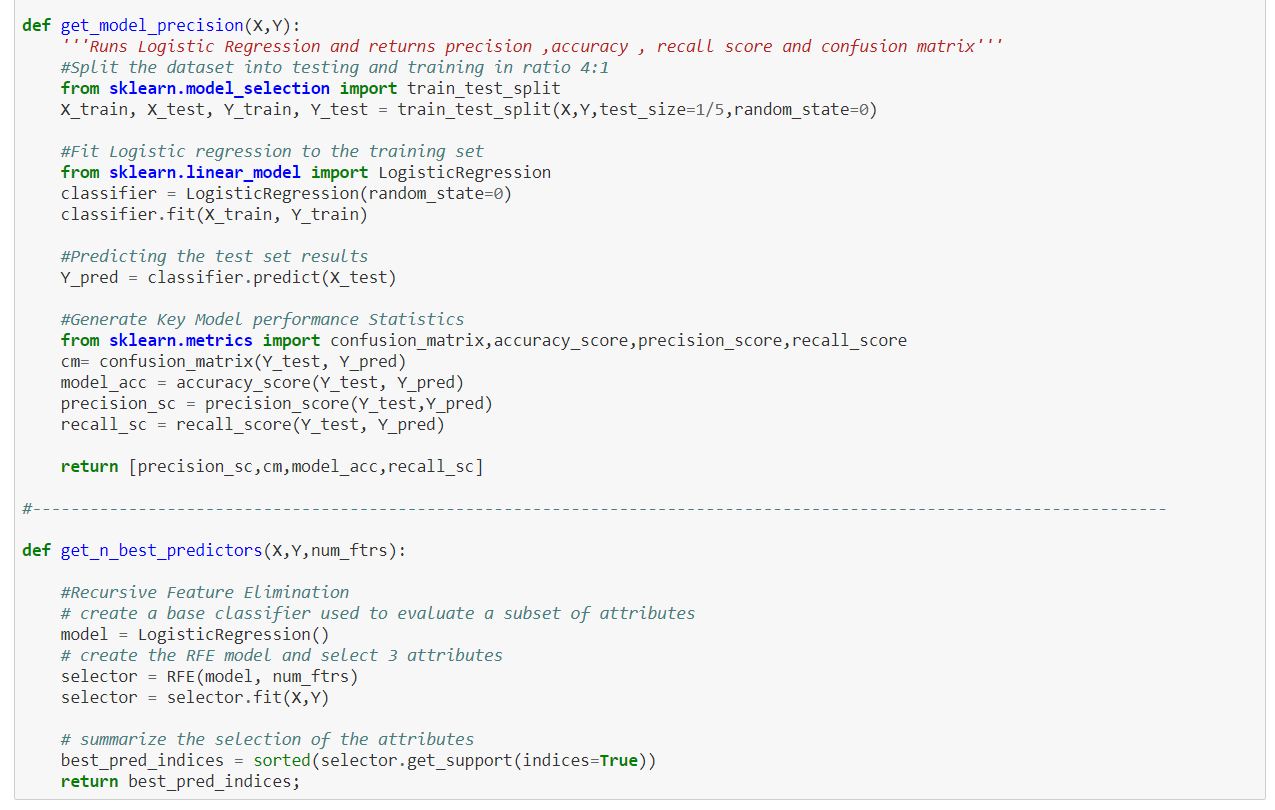
#### Convert data into Independent (X) and Dependent (Y) values



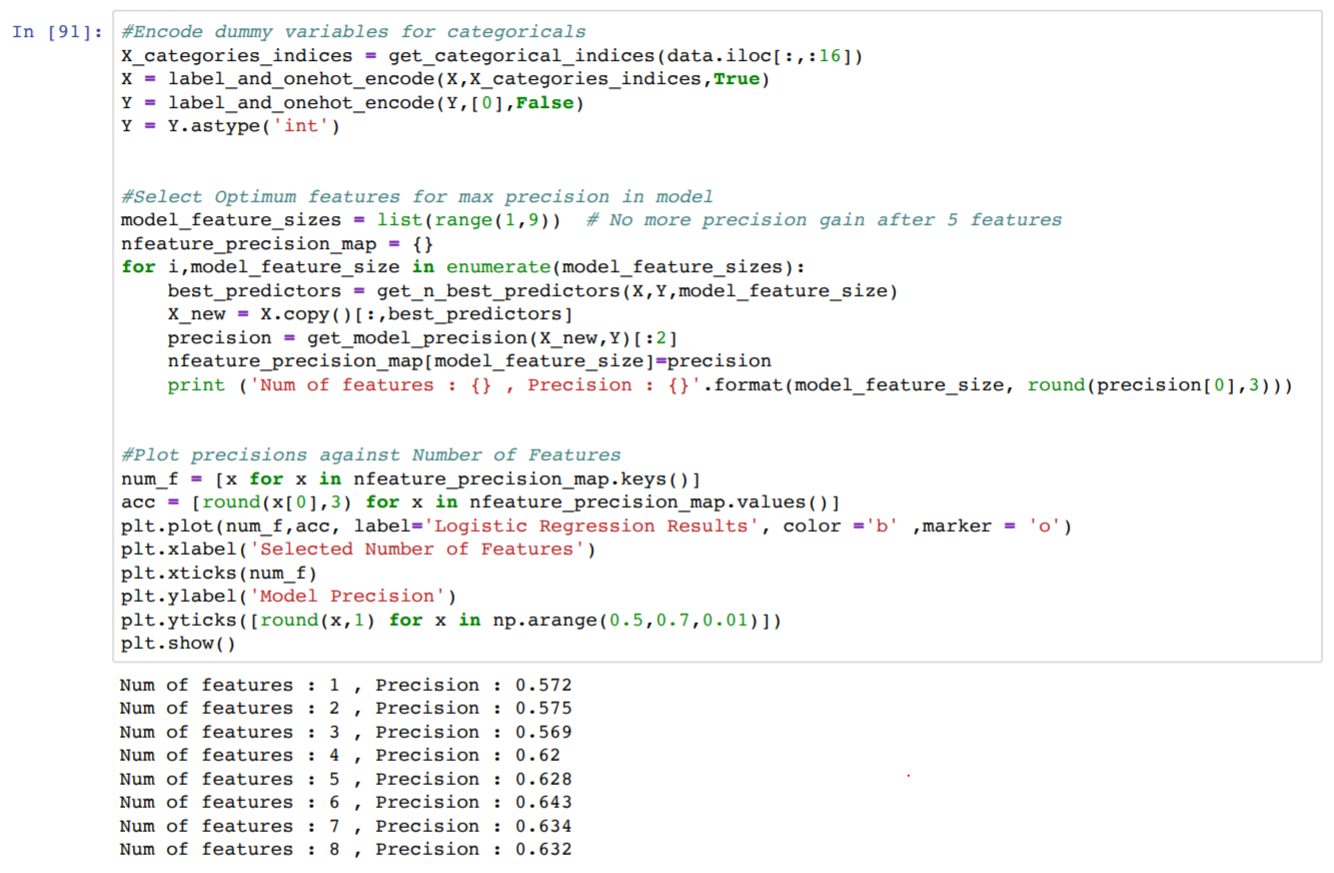
#### Define functions to assist in operations

We define functions below to help and assist in conducting logistic regression.



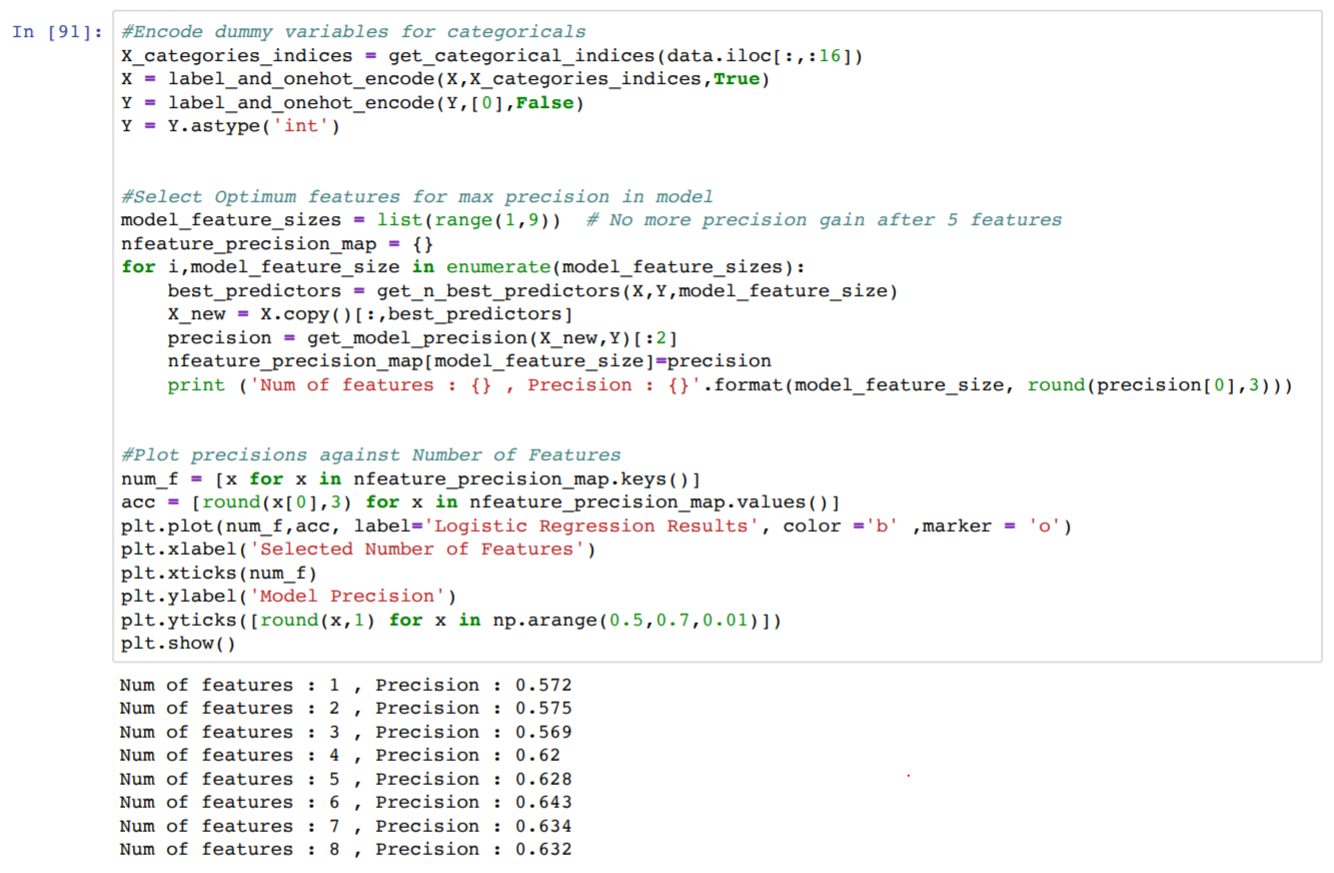


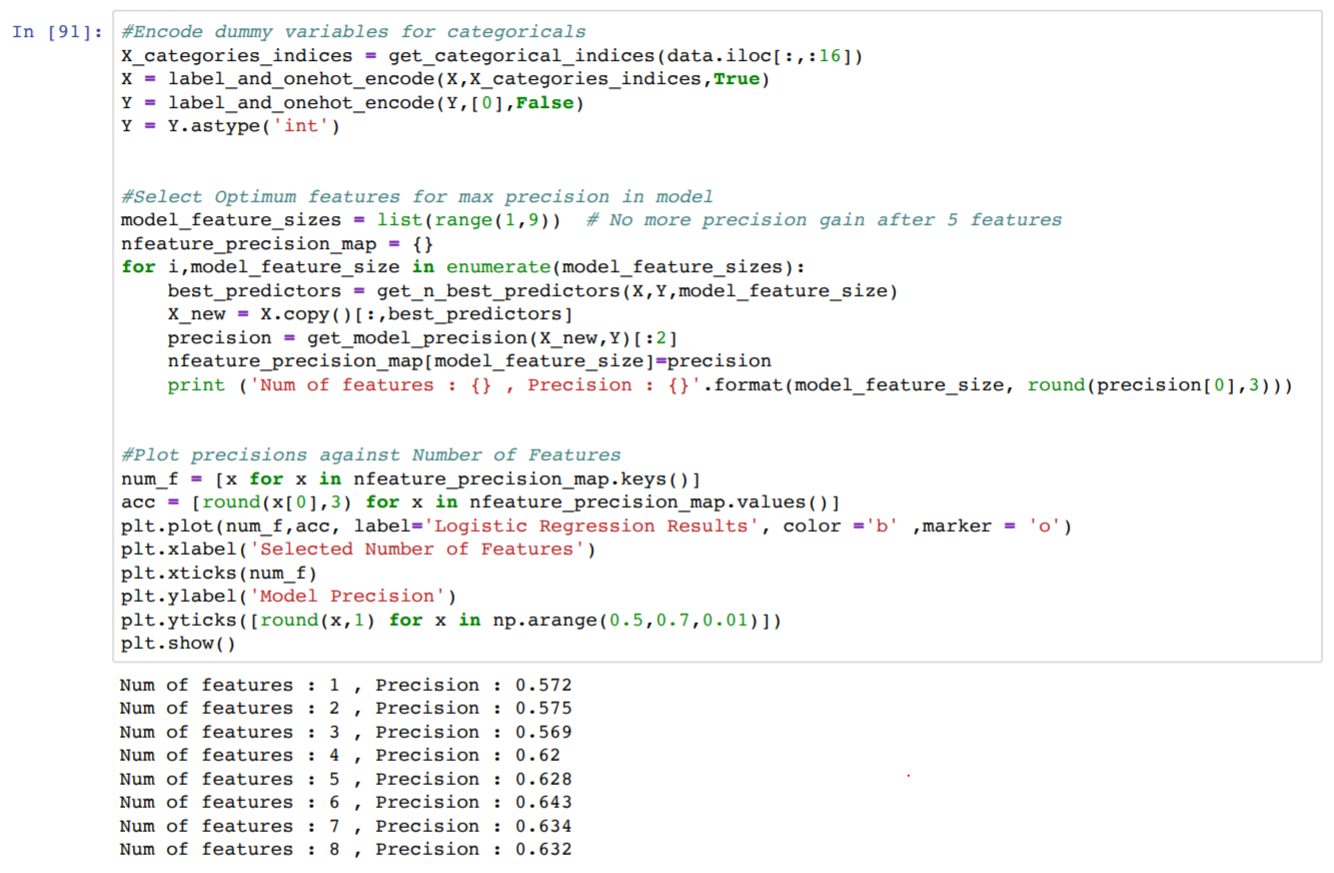
#### Encode dummy variables for categorical variables



#### Select optimum features for logistic regression model

We then run our functions we have created earlier to get the optimum features for the logistic regression model.

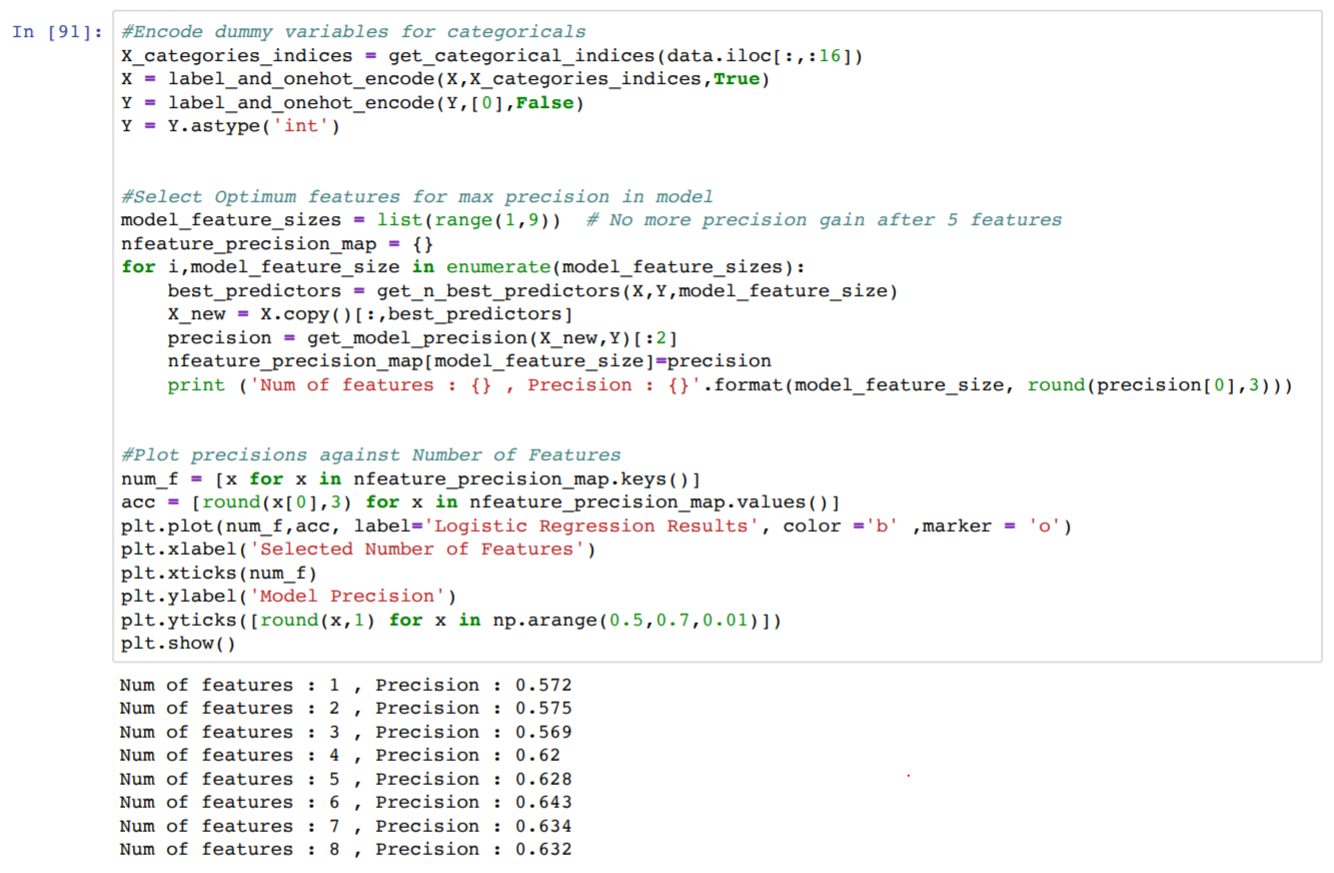


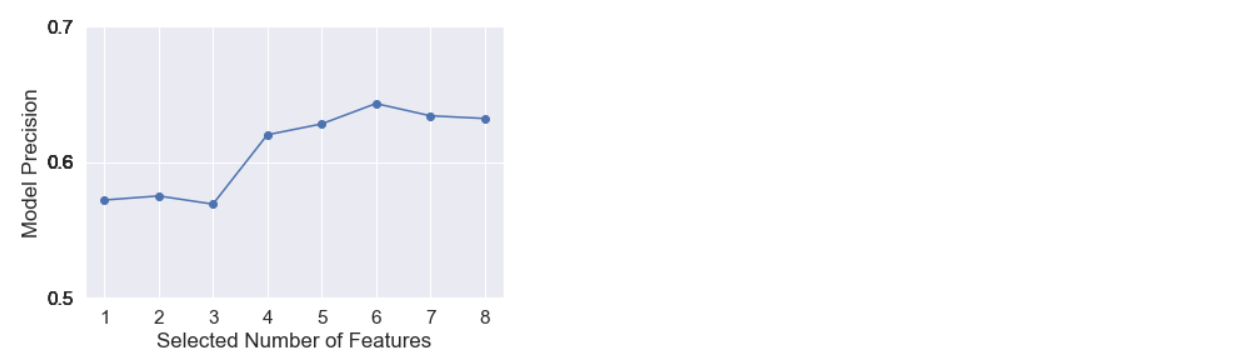


As we can see from the results, the optimum number of features is 6 at a precision of 64%.

#### Plot precision against number of features

For visual representation of precision levels against the number of features we will plot a line graph.





For the logistic regression model as well, we get an accuracy of 89% and precision of 64%.