LAMINA STUDIOS, LLC
DATA ANALYTICS INTERNSHIP

Logistic Regression using Python

Documentation

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

September 27, 2023

Table of Contents

Task Overview

For Data Analysts and Data Scientists, Python has many advantages. A huge range of open-source libraries make it an incredibly useful tool for any Data Analyst.

We have pandas, NumPy and Vaex for data analysis, Matplotlib, seaborn and Bokeh for visualisation, and TensorFlow, scikit-learn and PyTorch for machine learning applications.

With its (relatively) easy learning curve and versatility, it's no wonder that Python is one of the fastest-growing programming languages out there.

While there is a massive variety of sources for datasets, in many cases - particularly in enterprise businesses - data is going to be stored in a relational database. Relational databases are an extremely efficient, powerful and widely-used way to create, read, update and delete data of all kinds.

The most widely used relational database management systems (RDBMSs) - Oracle, MySQL, Microsoft SQL Server, PostgreSQL, IBM DB2 - all use the Structured Query Language (SQL) to access and make changes to the data.

In this task, I used Jupyter Notebook and Oracle MySQL to create a logistic regression using Python. This task also required the python libraries MySQL connector and pandas.

Abstract: The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Data Set Information: The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Attribute Information:

Bank client data:

- Age (numeric)
- Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
- Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)
- Education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
- Default: has credit in default? (categorical: 'no', 'yes', 'unknown')
- Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')
- Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

Related with the last contact of the current campaign:

- Contact: contact communication type (categorical: 'cellular','telephone')
- Month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
- Day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
- Duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Other attributes:

- Campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- Pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- Previous: number of contacts performed before this campaign and for this client (numeric)
- Poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Social and economic context attributes:

- Emp.var.rate: employment variation rate - quarterly indicator (numeric)
- Cons.price.idx: consumer price index - monthly indicator (numeric)
- Cons.conf.idx: consumer confidence index - monthly indicator (numeric)
- Euribor3m: euribor 3 month rate - daily indicator (numeric)
- Nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

- y - has the client subscribed a term deposit? (binary: 'yes', 'no')

Source:

- Dataset from : http://archive.ics.uci.edu/ml/datasets/Bank+Marketing#

Codes

The following are the steps with codes that was used for this task.

Step 1: Installation of Python and Oracle MySQL Server, Workbench, & Connector

Step 2: Installation of Jupyter on a terminal

```
python -m pip install jupyter
```

Step 3: Launching of Jupyter Notebook

```
Jupyter notebook
```

Step 4: Install MySQL Connector Python Library

```
!pip install mysql-connector-python
```

Step 5: Install Panda Python Library

```
!pip install pandas
```

Step 6: Importing the installed libraries

```
# Importing Data Analysis Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Step 7: Reading the file and displaying it

```
bank = pd.read_csv('../input/bank-additional-full.csv', sep = ';')
#Converting dependent variable categorical to dummy
y = pd.get_dummies(bank['y'], columns = ['y'], prefix = ['y'], drop_first = True)
bank.head()
```

Step 8: Taking a look at the information of csv file

```
# take a look at the type, number of columns, entries, null values etc..
bank.info()
# bank.isnull().any() # one way to search for null values
```

Step 9: Taking a look at the variables or columns

```
bank.columns
```

Step 10: Bank Client Data Analysis and Categorical Treatment

```
bank_client = bank.iloc[: , 0:7]
bank_client.head()
```

Step 11: Knowing the categorical variables of job, marital status, education, housing, loan

```
# knowing the categorical variables
print('Jobs:\n', bank_client['job'].unique())
print('Marital:\n', bank_client['marital'].unique())
print('Education:\n', bank_client['education'].unique())
```

```
print('Default:\n', bank_client['default'].unique())
print('Housing:\n', bank_client['housing'].unique())
print('Loan:\n', bank_client['loan'].unique())
```

Step 12: Trying to find some insights crossing those variables

```
#Trying to find some strange values or null values
print('Min age: ', bank_client['age'].max())
print('Max age: ', bank_client['age'].min())
print('Null Values: ', bank_client['age'].isnull().any())
```

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'age', data = bank_client)
ax.set_xlabel('Age', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Age Count Distribution', fontsize=15)
sns.despine()
```

```
fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize = (13, 5))
sns.boxplot(x = 'age', data = bank_client, orient = 'v', ax = ax1)
ax1.set_xlabel('People Age', fontsize=15)
ax1.set_ylabel('Age', fontsize=15)
ax1.set_title('Age Distribution', fontsize=15)
ax1.tick_params(labelsize=15)

sns.distplot(bank_client['age'], ax = ax2)
sns.despine(ax = ax2)
ax2.set_xlabel('Age', fontsize=15)
ax2.set_ylabel('Occurence', fontsize=15)
ax2.set_title('Age x Ocucurence', fontsize=15)
ax2.tick_params(labelsize=15)

plt.subplots_adjust(wspace=0.5)
plt.tight_layout()
```

Step 13: Calculating the quartiles

```
# Quartiles
print('1º Quartile: ', bank_client['age'].quantile(q = 0.25))
print('2º Quartile: ', bank_client['age'].quantile(q = 0.50))
print('3º Quartile: ', bank_client['age'].quantile(q = 0.75))
print('4º Quartile: ', bank_client['age'].quantile(q = 1.00))
#Calculate the outliers:
  # Interquartile range, IQR = Q3 - Q1
  # lower 1.5*IQR whisker = Q1 - 1.5 * IQR
  # Upper 1.5*IQR whisker = Q3 + 1.5 * IQR

print('Ages above: ', bank_client['age'].quantile(q = 0.75) +
            1.5*(bank_client['age'].quantile(q = 0.75) - bank_client['age'].quantile(q = 0.25)), 'are outliers')
```

Step 14: Number of outliers

```
print('Numerber of outliers: ', bank_client[bank_client['age'] > 69.6]['age'].count())
print('Number of clients: ', len(bank_client))
#Outliers in %
print('Outliers are:', round(bank_client[bank_client['age'] > 69.6]['age'].count()*100/len(bank_client),2), '%')
```

Step 15: Calculating mean, std, and cv

```
# Calculating some values to evaluete this independent variable
print('MEAN:', round(bank_client['age'].mean(), 1))
# A low standard deviation indicates that the data points tend to be close to the mean or expected value
# A high standard deviation indicates that the data points are scattered
print('STD :', round(bank_client['age'].std(), 1))
# I thing the best way to give a precisly insight abou dispersion is using the CV (coefficient variation) (STD/MEAN)*100
#    cv < 15%, low dispersion
#    cv > 30%, high dispersion
print('CV  :',round(bank_client['age'].std()*100/bank_client['age'].mean(), 1), ', High middle dispersion')
```

Step 16: Data Visualization for jobs, marital, education, default, housing, and loan

```
# What kind of jobs clients this bank have, if you cross jobs with default, loan or housing, there is no relation
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'job', data = bank_client)
ax.set_xlabel('Job', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Age Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```

```
# What kind of 'marital clients' this bank have, if you cross marital with default, loan or housing, there is no relation
fig, ax = plt.subplots()
fig.set_size_inches(10, 5)
sns.countplot(x = 'marital', data = bank_client)
ax.set_xlabel('Marital', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Age Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```

```
# What kind of 'education clients this bank have, if you cross education with default, loan or housing, there is no relation
fig, ax = plt.subplots()
fig.set_size_inches(20, 5)
sns.countplot(x = 'education', data = bank_client)
ax.set_xlabel('Education', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Education Count Distribution', fontsize=15)
ax.tick_params(labelsize=15)
sns.despine()
```

```
# Default, has credit in default ?
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (20,8))
sns.countplot(x = 'default', data = bank_client, ax = ax1, order = ['no', 'unknown', 'yes'])
ax1.set_title('Default', fontsize=15)
ax1.set_xlabel('')
ax1.set_ylabel('Count', fontsize=15)
ax1.tick_params(labelsize=15)
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

```
# Housing, has housing loan ?
sns.countplot(x = 'housing', data = bank_client, ax = ax2, order = ['no', 'unknown', 'yes'])
ax2.set_title('Housing', fontsize=15)
ax2.set_xlabel('')
ax2.set_ylabel('Count', fontsize=15)
ax2.tick_params(labelsize=15)

# Loan, has personal loan ?
sns.countplot(x = 'loan', data = bank_client, ax = ax3, order = ['no', 'unknown', 'yes'])
ax3.set_title('Loan', fontsize=15)
ax3.set_xlabel('')
ax3.set_ylabel('Count', fontsize=15)
ax3.tick_params(labelsize=15)

plt.subplots_adjust(wspace=0.25)
```

```
print('Default:\n No credit in default:'     , bank_client[bank_client['default'] == 'no']['age'].count(),
      '\n Unknown credit in default:', bank_client[bank_client['default'] == 'unknown']['age'].count(),
      '\n Yes to credit in default:' , bank_client[bank_client['default'] == 'yes']['age'].count())
```

```
print('Housing:\n No housing in loan:'     , bank_client[bank_client['housing'] == 'no']['age'].count(),
      '\n Unknown housing in loan:', bank_client[bank_client['housing'] == 'unknown']['age'].count(),
      '\n Yes to housing in loan:' , bank_client[bank_client['housing'] == 'yes']['age'].count())
```

```
print('Housing:\n No to personal loan:'     , bank_client[bank_client['loan'] == 'no']['age'].count(),
      '\n Unknown to personal loan:', bank_client[bank_client['loan'] == 'unknown']['age'].count(),
      '\n Yes to personal loan:'    , bank_client[bank_client['loan'] == 'yes']   ['age'].count())
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Step 17: Bank categorical treatment: Jobs, Marital, Education, Default, Housing, Loan. Converting to continuous due the feature scaling will be applied later

```
# Label encoder order is alphabetical
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_client['job']      = labelencoder_X.fit_transform(bank_client['job'])
bank_client['marital']  = labelencoder_X.fit_transform(bank_client['marital'])
bank_client['education']= labelencoder_X.fit_transform(bank_client['education'])
bank_client['default']  = labelencoder_X.fit_transform(bank_client['default'])
bank_client['housing']  = labelencoder_X.fit_transform(bank_client['housing'])
bank_client['loan']     = labelencoder_X.fit_transform(bank_client['loan'])
```

```
#function to creat group of ages, this helps because we have 78 differente values here
def age(dataframe):
    dataframe.loc[dataframe['age'] <= 32, 'age'] = 1
    dataframe.loc[(dataframe['age'] > 32) & (dataframe['age'] <= 47), 'age'] = 2
    dataframe.loc[(dataframe['age'] > 47) & (dataframe['age'] <= 70), 'age'] = 3
    dataframe.loc[(dataframe['age'] > 70) & (dataframe['age'] <= 98), 'age'] = 4

    return dataframe

age(bank_client);
```

```
bank_client.head()
```

```
print(bank_client.shape)
bank_client.head()
```

Step 18: Related with the last contact of the current campaign. Treat categorical, see those values and group continuous variables if necessary

```
# Slicing DataFrame to treat separately, make things more easy
bank_related = bank.iloc[: , 7:11]
bank_related.head()
```

```
bank_related.isnull().any()
```

```
print("Kind of Contact: \n", bank_related['contact'].unique())
print("\nWhich monthis this campaing work: \n", bank_related['month'].unique())
print("\nWhich days of week this campaing work: \n", bank_related['day_of_week'].unique())
```

```
fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize = (13, 5))
sns.boxplot(x = 'duration', data = bank_related, orient = 'v', ax = ax1)
```

```
ax1.set_xlabel('Calls', fontsize=10)
ax1.set_ylabel('Duration', fontsize=10)
ax1.set_title('Calls Distribution', fontsize=10)
ax1.tick_params(labelsize=10)

sns.distplot(bank_related['duration'], ax = ax2)
sns.despine(ax = ax2)
ax2.set_xlabel('Duration Calls', fontsize=10)
ax2.set_ylabel('Occurence', fontsize=10)
ax2.set_title('Duration x Ocucurence', fontsize=10)
ax2.tick_params(labelsize=10)

plt.subplots_adjust(wspace=0.5)
plt.tight_layout()
```

```
print("Max duration  call in minutes:  ", round((bank_related['duration'].max()/60),1))
print("Min duration  call in minutes:   ", round((bank_related['duration'].min()/60),1))
print("Mean duration call in minutes:   ", round((bank_related['duration'].mean()/60),1))
print("STD duration  call in minutes:   ", round((bank_related['duration'].std()/60),1))
# Std close to the mean means that the data values are close to the mean
```

Step 19: Quartiles and outliers

```
# Quartiles
print('1º Quartile: ', bank_related['duration'].quantile(q = 0.25))
print('2º Quartile: ', bank_related['duration'].quantile(q = 0.50))
print('3º Quartile: ', bank_related['duration'].quantile(q = 0.75))
print('4º Quartile: ', bank_related['duration'].quantile(q = 1.00))
#Calculate the outliers:
  # Interquartile range, IQR = Q3 - Q1
  # lower 1.5*IQR whisker = Q1 - 1.5 * IQR
  # Upper 1.5*IQR whisker = Q3 + 1.5 * IQR

print('Duration calls above: ', bank_related['duration'].quantile(q = 0.75) +
                      1.5*(bank_related['duration'].quantile(q = 0.75) -
bank_related['duration'].quantile(q = 0.25)), 'are outliers')
```

```
print('Numerber of outliers: ', bank_related[bank_related['duration'] >
644.5]['duration'].count())
print('Number of clients: ', len(bank_related))
#Outliers in %
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

```
print('Outliers are:', round(bank_related[bank_related['duration'] > 644.5]['duration'].count()*100/len(bank_related),2), '%')
```

Step 20: Check if call duration is 0

```
# Look, if the call duration is iqual to 0, then is obviously that this person didn't subscribed,
# THIS LINES NEED TO BE DELETED LATER
bank[(bank['duration'] == 0)]
```

Step 21: Contact, Month, Day of week

```
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (15,6))
sns.countplot(bank_related['contact'], ax = ax1)
ax1.set_xlabel('Contact', fontsize = 10)
ax1.set_ylabel('Count', fontsize = 10)
ax1.set_title('Contact Counts')
ax1.tick_params(labelsize=10)

sns.countplot(bank_related['month'], ax = ax2, order = ['mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec'])
ax2.set_xlabel('Months', fontsize = 10)
ax2.set_ylabel('')
ax2.set_title('Months Counts')
ax2.tick_params(labelsize=10)

sns.countplot(bank_related['day_of_week'], ax = ax3)
ax3.set_xlabel('Day of Week', fontsize = 10)
ax3.set_ylabel('')
ax3.set_title('Day of Week Counts')
ax3.tick_params(labelsize=10)

plt.subplots_adjust(wspace=0.25)
```

```
print('Ages above: ', bank_related['duration'].quantile(q = 0.75) +
        1.5*(bank_related['duration'].quantile(q = 0.75) - bank_related['duration'].quantile(q = 0.25)), 'are outliers')
```

```
bank_related[bank_related['duration'] > 640].count()
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Step 22: Contact, Month, Day of week Treatment

```
# Label encoder order is alphabetical
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_related['contact']     = labelencoder_X.fit_transform(bank_related['contact'])
bank_related['month']       = labelencoder_X.fit_transform(bank_related['month'])
bank_related['day_of_week'] = labelencoder_X.fit_transform(bank_related['day_of_week'])
```

```
bank_related.head()
```

```
def duration(data):

    data.loc[data['duration'] <= 102, 'duration'] = 1
    data.loc[(data['duration'] > 102) & (data['duration'] <= 180)  , 'duration']    = 2
    data.loc[(data['duration'] > 180) & (data['duration'] <= 319)  , 'duration']   = 3
    data.loc[(data['duration'] > 319) & (data['duration'] <= 644.5), 'duration'] = 4
    data.loc[data['duration']  > 644.5, 'duration'] = 5

    return data
duration(bank_related);
```

```
bank_related.head()
```

Step 23: Socio, economic, and other attributes

```
bank_se = bank.loc[: , ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 
'nr.employed']]
bank_se.head()
```

```
bank_o = bank.loc[: , ['campaign', 'pdays','previous', 'poutcome']]
bank_o.head()
```

```
bank_o['poutcome'].unique()
```

```
bank_o['poutcome'].replace(['nonexistent', 'failure', 'success'], [1,2,3], inplace  = True)
```

Step 24: Model

```
bank_final= pd.concat([bank_client, bank_related, bank_se, bank_o], axis = 1)
bank_final = bank_final[['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
                     'contact', 'month', 'day_of_week', 'duration', 'emp.var.rate', 'cons.price.idx',
                     'cons.conf.idx', 'euribor3m', 'nr.employed', 'campaign', 'pdays', 'previous',
'poutcome']]
bank_final.shape
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Step 25: Train test split the data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(bank_final, y, test_size = 0.1942313295, random_state = 101)

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

```
X_train.head()
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Step 26: Training and predicting the logistic regression model

```
from sklearn.linear_model import LogisticRegression
# Create the logistic regression model
logmodel = LogisticRegression()

# Train the model
logmodel.fit(X_train, y_train)

# Make predictions on the test data
logpred = logmodel.predict(X_test)

# Evaluate the model
print("Confusion Matrix:\n", confusion_matrix(y_test, logpred))
print("\nAccuracy:\n", round(accuracy_score(y_test, logpred), 2) * 100)

# Cross-validate the model
LOGCV = cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean()
```

Step 27: Confusion Matri for logistic regression

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the confusion matrix
cm = confusion_matrix(y_test, logpred)

# Plot the confusion matrix as a heatmap
sns.heatmap(cm, annot=True, fmt='.0f')

# Set the title and labels
plt.title('Confusion Matrix for Logistic Regression')
plt.xlabel('Predicted Class')
plt.ylabel('Actual Class')

# Show the plot
plt.show()
```

Step 28: Print the cross-validation score

```
# Print the cross-validation score
print(LOGCV)
```

Step 29: Showing ROC Curve for logistic regression

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Calculate the ROC curve and AUC score for the Logistic Regression model
probs = logmodel.predict_proba(X_test)
preds = probs[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
```

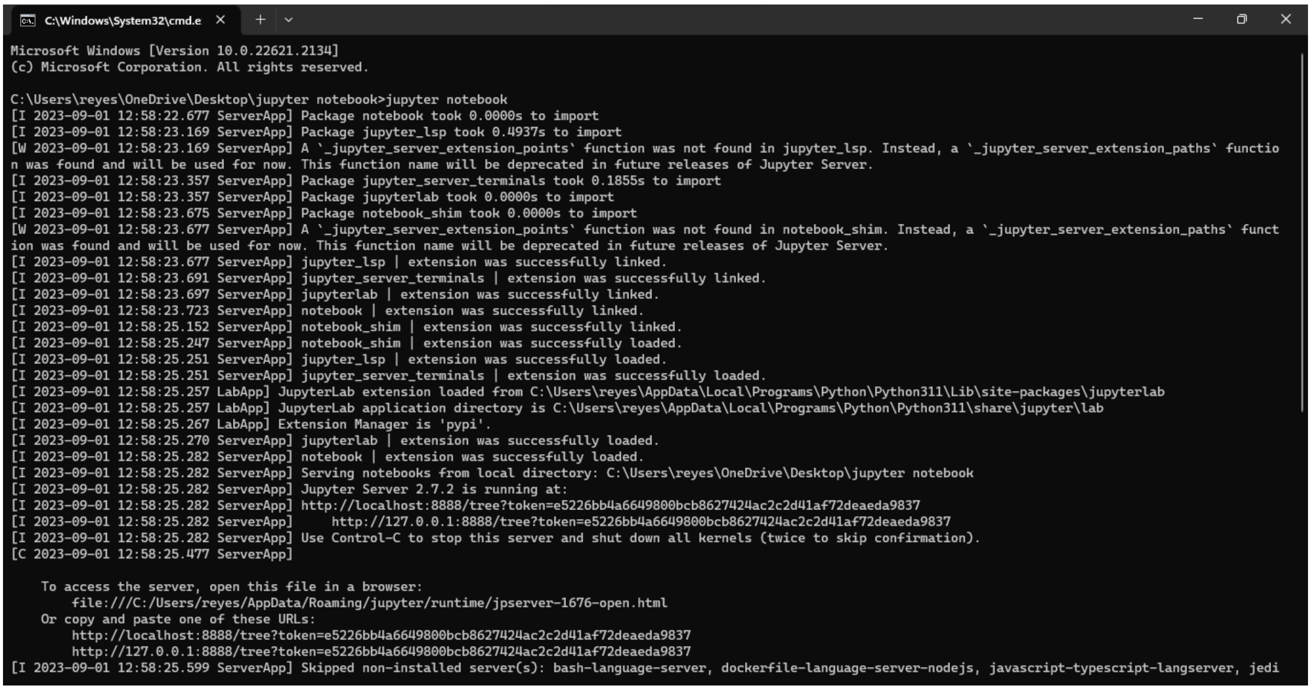
```
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Logistic Regression')
plt.legend(loc='lower right')
plt.show()
```

Step 30: Classification report for logistic regression

```
from sklearn.metrics import classification_report

# Calculate the classification report for the Logistic Regression model
report = classification_report(y_test, logpred)

# Print the classification report
print(report)
```

Screenshots of Work

Figure 1. Launching of Jupyter Notebook



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\reyes\OneDrive\Desktop\jupyter notebook>jupyter notebook
[I 2023-09-01 12:58:22.677 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-01 12:58:23.169 ServerApp] Package jupyter_lsp took 0.4937s to import
[W 2023-09-01 12:58:23.169 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_paths` functio
n was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-01 12:58:23.357 ServerApp] Package jupyter_server_terminals took 0.1855s to import
[I 2023-09-01 12:58:23.357 ServerApp] Package jupyterlab took 0.0000s to import
[I 2023-09-01 12:58:23.675 ServerApp] Package notebook_shim took 0.0000s to import
[W 2023-09-01 12:58:23.677 ServerApp] A `_jupyter_server_extension_points` function was not found in notebook_shim. Instead, a `_jupyter_server_extension_paths` funct
ion was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-01 12:58:23.677 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2023-09-01 12:58:23.691 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2023-09-01 12:58:23.697 ServerApp] jupyterlab | extension was successfully linked.
[I 2023-09-01 12:58:23.723 ServerApp] notebook | extension was successfully linked.
[I 2023-09-01 12:58:25.152 ServerApp] notebook_shim | extension was successfully linked.
[I 2023-09-01 12:58:25.247 ServerApp] notebook_shim | extension was successfully loaded.
[I 2023-09-01 12:58:25.251 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2023-09-01 12:58:25.251 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-09-01 12:58:25.257 LabApp] JupyterLab extension loaded from C:\Users\reyes\AppData\Local\Programs\Python\Python311\Lib\site-packages\jupyterlab
[I 2023-09-01 12:58:25.257 LabApp] JupyterLab application directory is C:\Users\reyes\AppData\Local\Programs\Python\Python311\share\jupyter\lab
[I 2023-09-01 12:58:25.267 LabApp] Extension Manager is 'pypi'.
[I 2023-09-01 12:58:25.270 ServerApp] jupyterlab | extension was successfully loaded.
[I 2023-09-01 12:58:25.282 ServerApp] notebook | extension was successfully loaded.
[I 2023-09-01 12:58:25.282 ServerApp] Serving notebooks from local directory: C:\Users\reyes\OneDrive\Desktop\jupyter notebook
[I 2023-09-01 12:58:25.282 ServerApp] Jupyter Server 2.7.2 is running at:
[I 2023-09-01 12:58:25.282 ServerApp] http://localhost:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
[I 2023-09-01 12:58:25.282 ServerApp]     http://127.0.0.1:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
[I 2023-09-01 12:58:25.282 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2023-09-01 12:58:25.477 ServerApp]

    To access the server, open this file in a browser:
        file:///C:/Users/reyes/AppData/Roaming/jupyter/runtime/jpserver-1676-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
        http://127.0.0.1:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
[I 2023-09-01 12:58:25.599 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi
```

Figure 2. Jupyter Notebook localhost

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 3. Installing the MySQL Connector Python Library

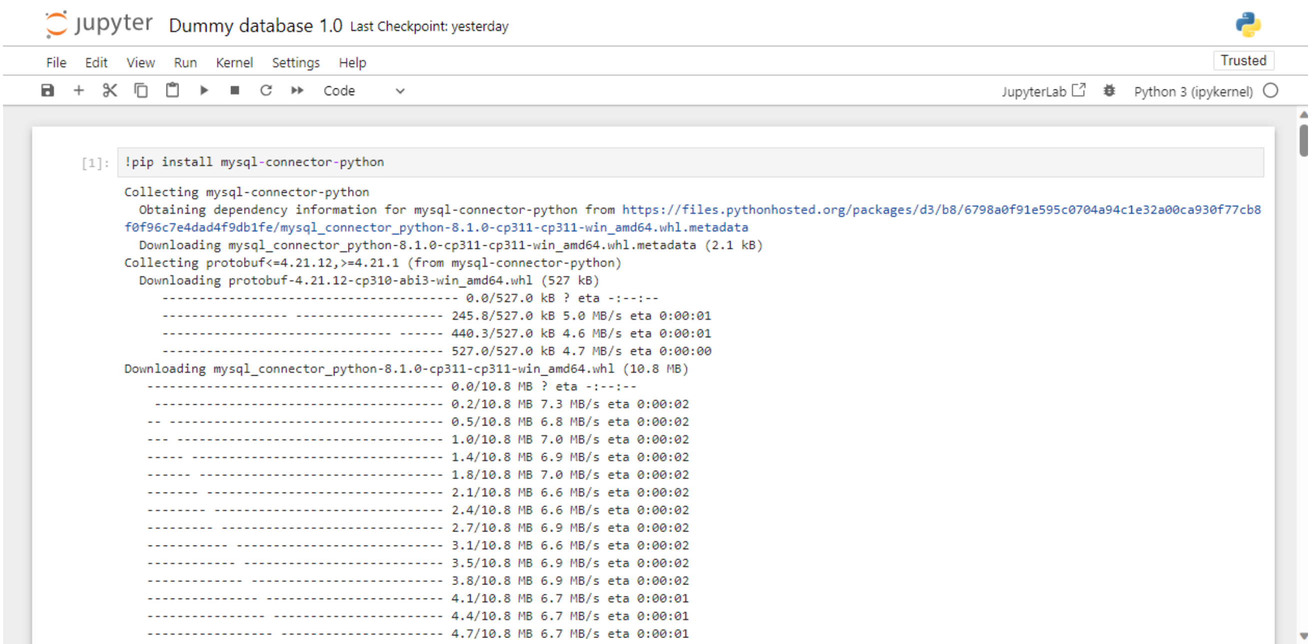
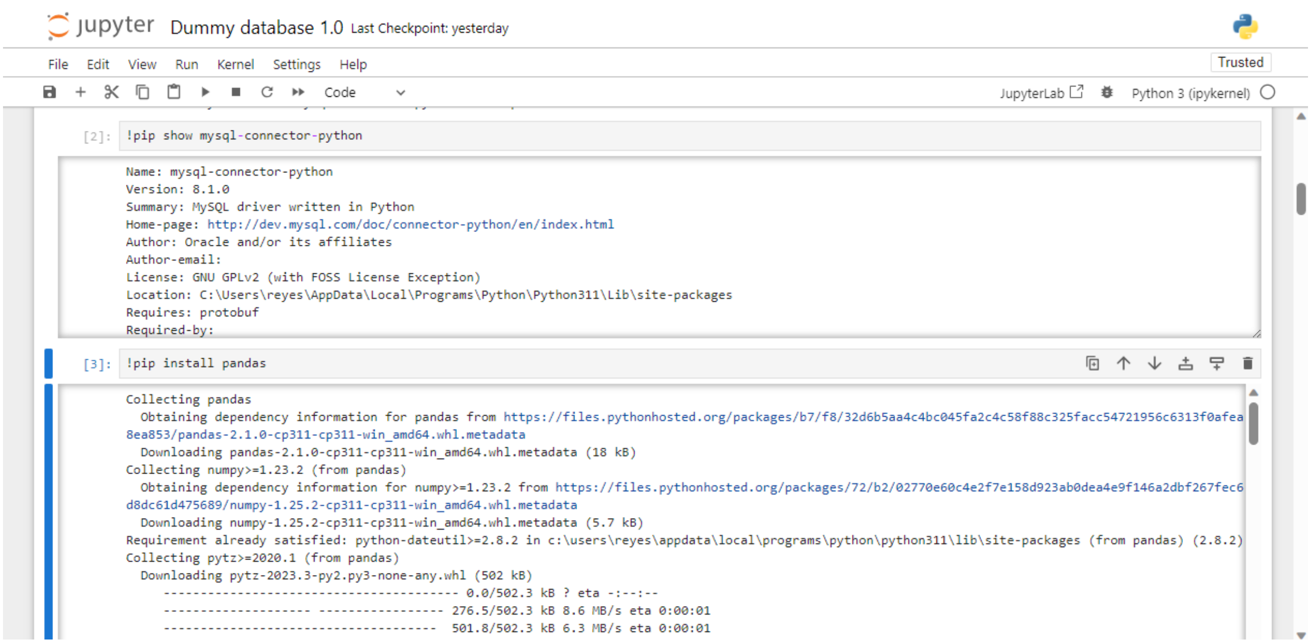
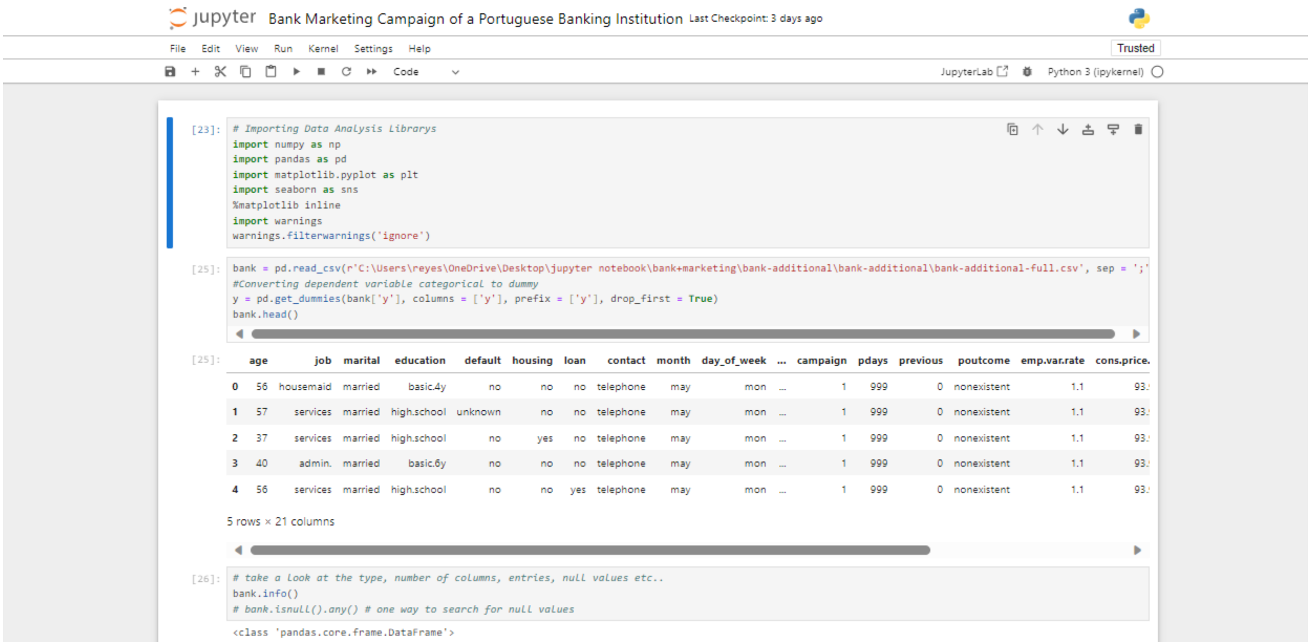


Figure 4. Installing the Pandas Python Library



Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 5. Importing the installed libraries



jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[23]:
```
# Importing Data Analysis Librarys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

[25]:
```
bank = pd.read_csv(r'C:\Users\reyes\OneDrive\Desktop\jupyter notebook\bank+marketing\bank-additional\bank-additional\bank-additional-full.csv', sep = ';'
#Converting dependent variable categorical to dummy
y = pd.get_dummies(bank['y'], columns = ['y'], prefix = ['y'], drop_first = True)
bank.head()
```

[25]:

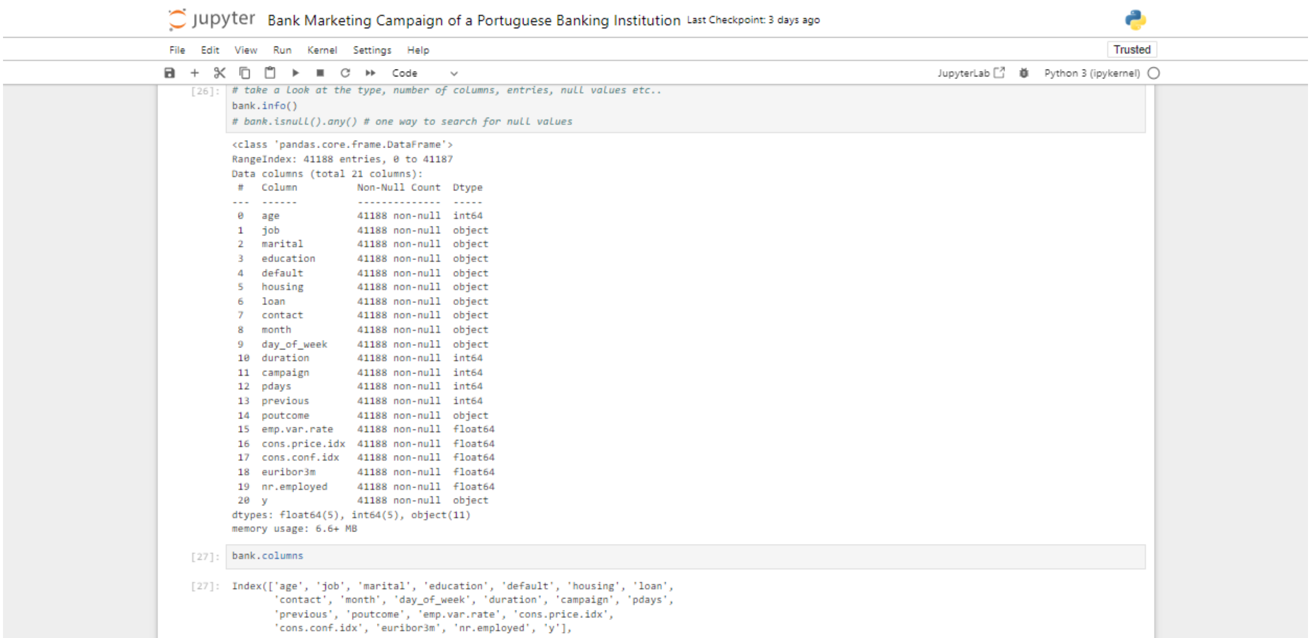
	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.

5 rows × 21 columns

[26]:
```
# take a look at the type, number of columns, entries, null values etc..
bank.info()
# bank.isnull().any() # one way to search for null values
```

```
<class 'pandas.core.frame.DataFrame'>
```

Figure 6. Information and columns of the csv file



jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

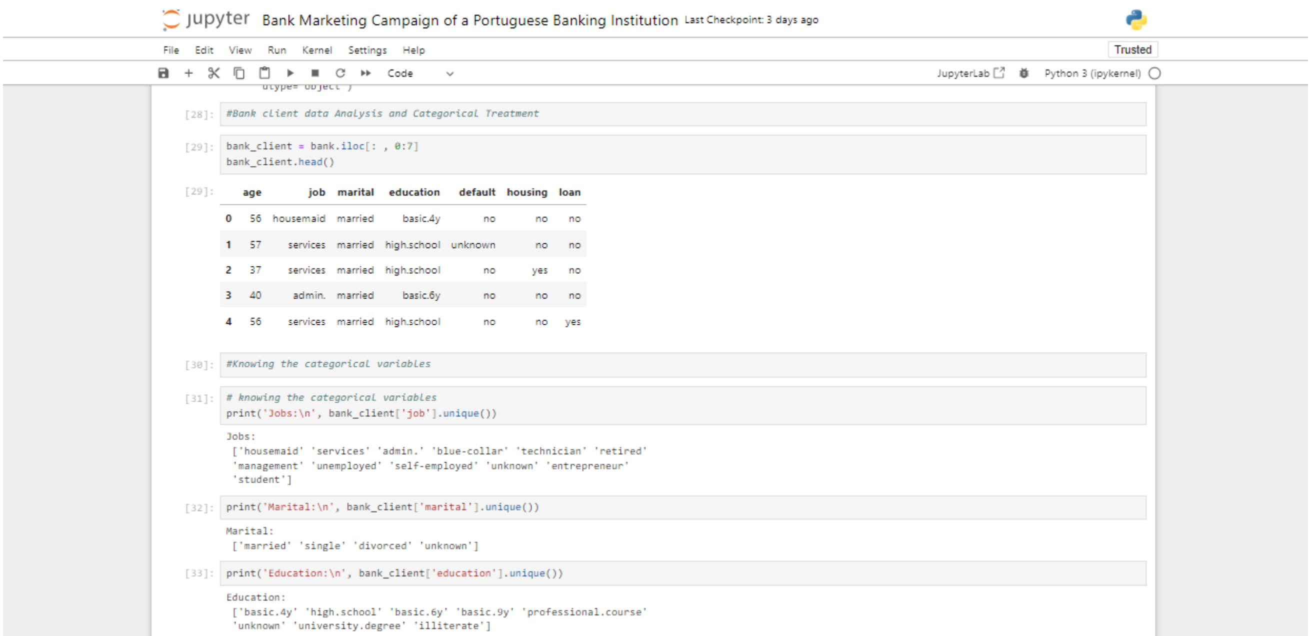
[26]:
```
# take a look at the type, number of columns, entries, null values etc..
bank.info()
# bank.isnull().any() # one way to search for null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

[27]:
```
bank.columns
```

[27]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
```

Figure 7. Bank Client data analysis and treatment



jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

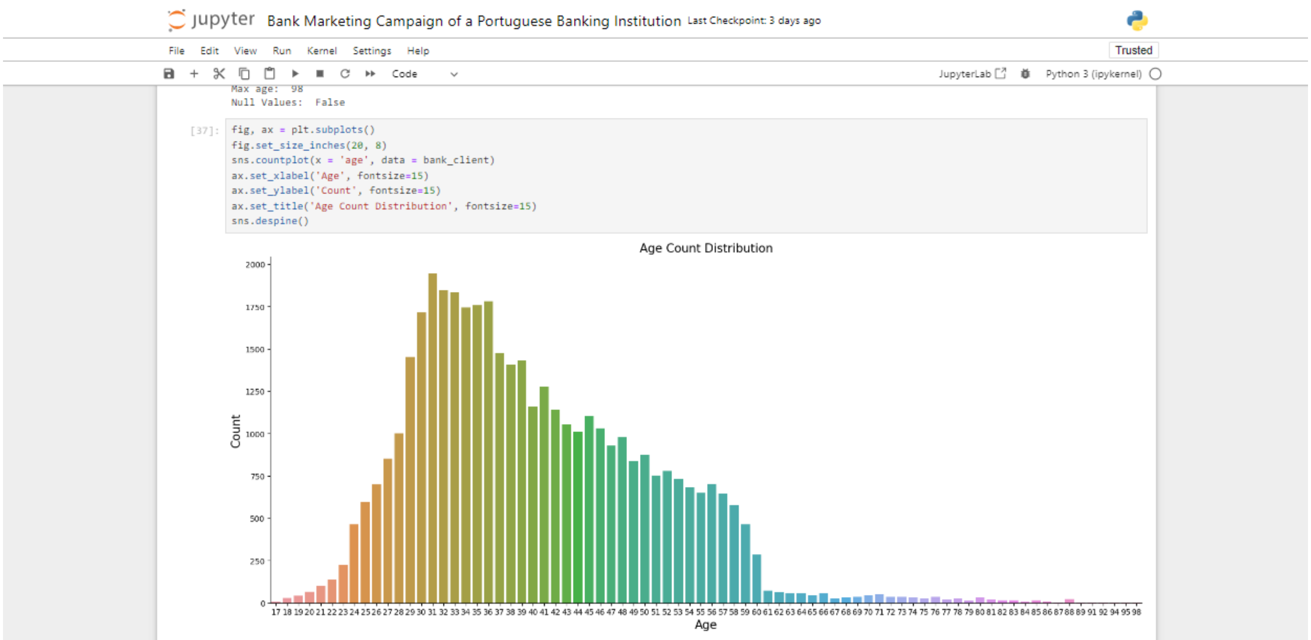
```
[28]: #Bank client data Analysis and Categorical Treatment

[29]: bank_client = bank.iloc[: , 0:7]
      bank_client.head()
```

[29]:

	age	job	marital	education	default	housing	loan
0	56	housemaid	married	basic.4y	no	no	no
1	57	services	married	high.school	unknown	no	no
2	37	services	married	high.school	no	yes	no
3	40	admin.	married	basic.6y	no	no	no
4	56	services	married	high.school	no	no	yes

```
[30]: #Knowing the categorical variables

[31]: # knowing the categorical variables
      print('Jobs:\n', bank_client['job'].unique())

Jobs:
 ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']

[32]: print('Marital:\n', bank_client['marital'].unique())

Marital:
 ['married' 'single' 'divorced' 'unknown']

[33]: print('Education:\n', bank_client['education'].unique())

Education:
 ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']
```

Figure 8. Age count distribution



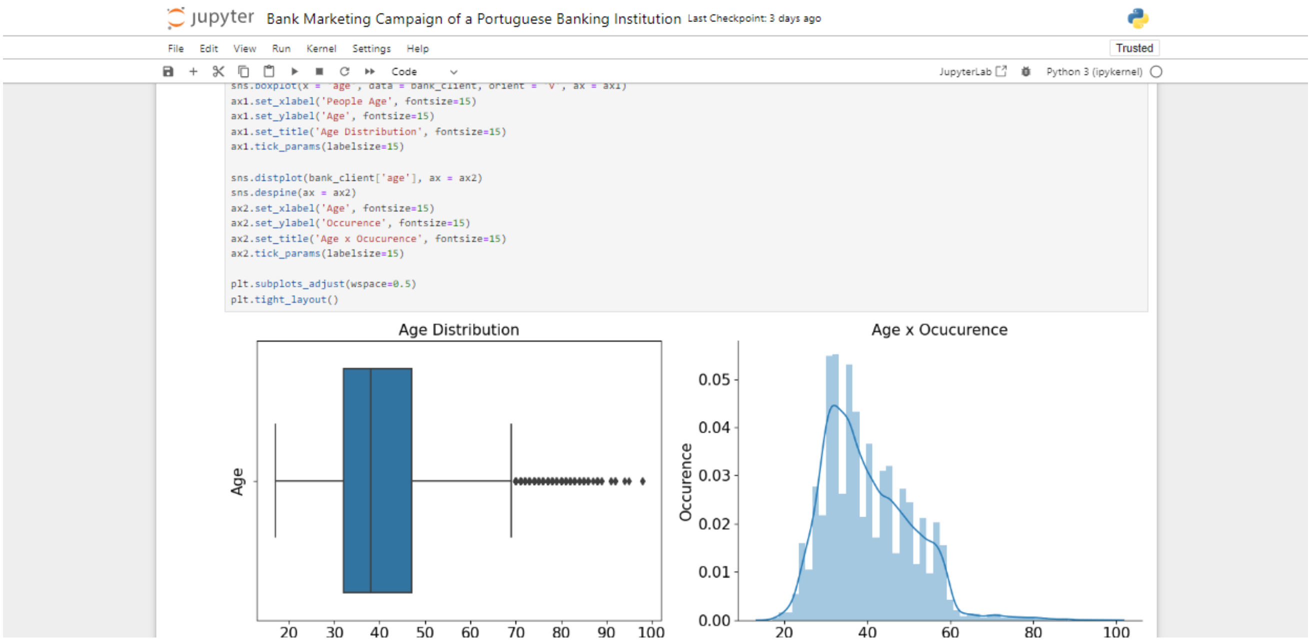
jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

```
Max age:  98
Null Values:  False

[37]: fig, ax = plt.subplots()
      fig.set_size_inches(20, 8)
      sns.countplot(x = 'age', data = bank_client)
      ax.set_xlabel('Age', fontsize=15)
      ax.set_ylabel('Count', fontsize=15)
      ax.set_title('Age Count Distribution', fontsize=15)
      sns.despine()
```

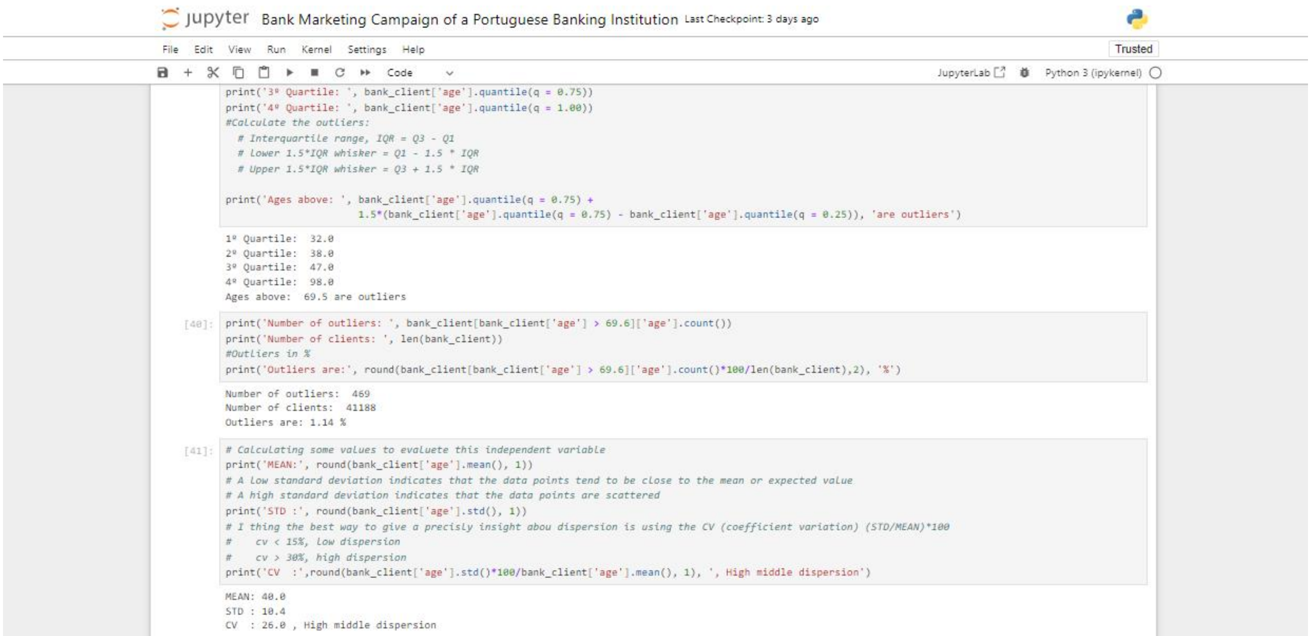

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 9. Age distribution age occurrence plot




```
sns.boxplot(x = 'age', data = bank_client, orient = 'v', ax = ax1)
ax1.set_xlabel('People Age', fontsize=15)
ax1.set_ylabel('Age', fontsize=15)
ax1.set_title('Age Distribution', fontsize=15)
ax1.tick_params(labelsize=15)

sns.distplot(bank_client['age'], ax = ax2)
sns.despine(ax = ax2)
ax2.set_xlabel('Age', fontsize=15)
ax2.set_ylabel('Occurence', fontsize=15)
ax2.set_title('Age x Ocucurence', fontsize=15)
ax2.tick_params(labelsize=15)

plt.subplots_adjust(wspace=0.5)
plt.tight_layout()
```

Figure 10. Quartiles, outliers, mean, STD, and CV




```
print('3º Quartile: ', bank_client['age'].quantile(q = 0.75))
print('4º Quartile: ', bank_client['age'].quantile(q = 1.00))
#Calculate the outliers:
  # Interquartile range, IQR = Q3 - Q1
  # lower 1.5*IQR whisker = Q1 - 1.5 * IQR
  # Upper 1.5*IQR whisker = Q3 + 1.5 * IQR

print('Ages above: ', bank_client['age'].quantile(q = 0.75) +
                      1.5*(bank_client['age'].quantile(q = 0.75) - bank_client['age'].quantile(q = 0.25)), 'are outliers')
```

```
1º Quartile:  32.0
2º Quartile:  38.0
3º Quartile:  47.0
4º Quartile:  98.0
Ages above:  69.5 are outliers
```

```
[40]: print('Number of outliers: ', bank_client[bank_client['age'] > 69.6]['age'].count())
print('Number of clients: ', len(bank_client))
#Outliers in %
print('Outliers are:', round(bank_client[bank_client['age'] > 69.6]['age'].count()*100/len(bank_client),2), '%')
```

```
Number of outliers:  469
Number of clients:  41188
Outliers are: 1.14 %
```

```
[41]: # Calculating some values to evaluete this independent variable
print('MEAN:', round(bank_client['age'].mean(), 1))
# A low standard deviation indicates that the data points tend to be close to the mean or expected value
# A high standard deviation indicates that the data points are scattered
print('STD :', round(bank_client['age'].std(), 1))
# I thing the best way to give a precisly insight abou dispersion is using the CV (coefficient variation) (STD/MEAN)*100
#    cv < 15%, low dispersion
#    cv > 30%, high dispersion
print('CV  :',round(bank_client['age'].std()*100/bank_client['age'].mean(), 1), ', High middle dispersion')
```

```
MEAN: 40.0
STD : 10.4
CV  : 26.0 , High middle dispersion
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 11. Job count distribution

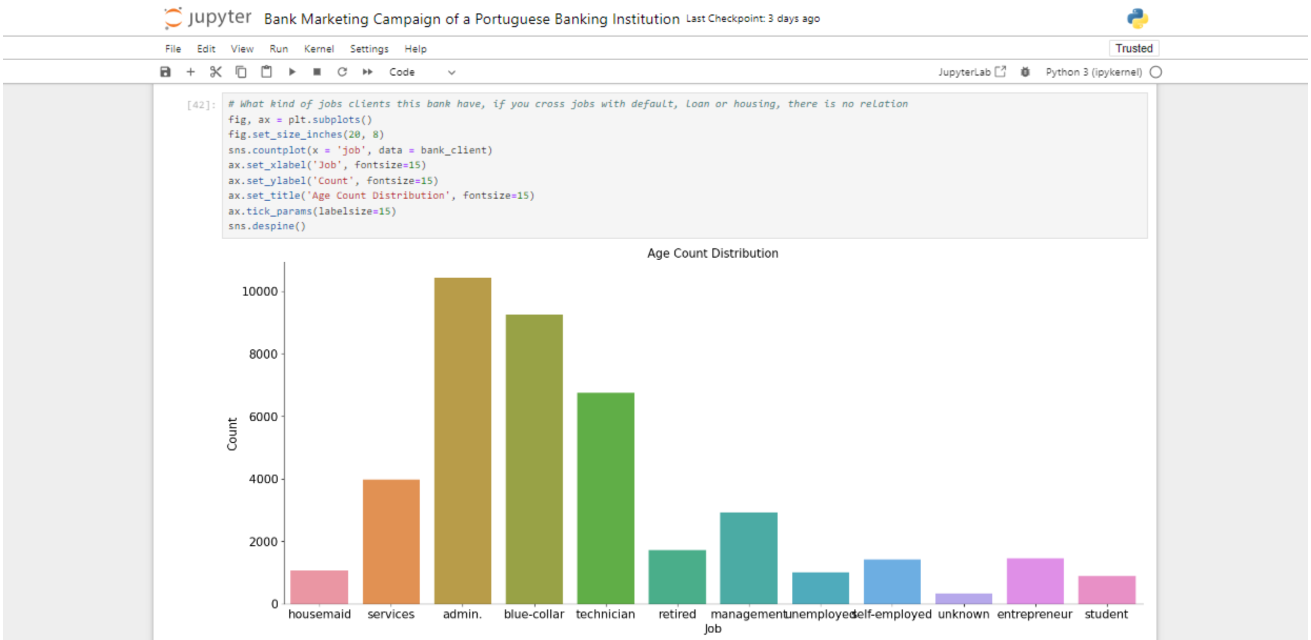
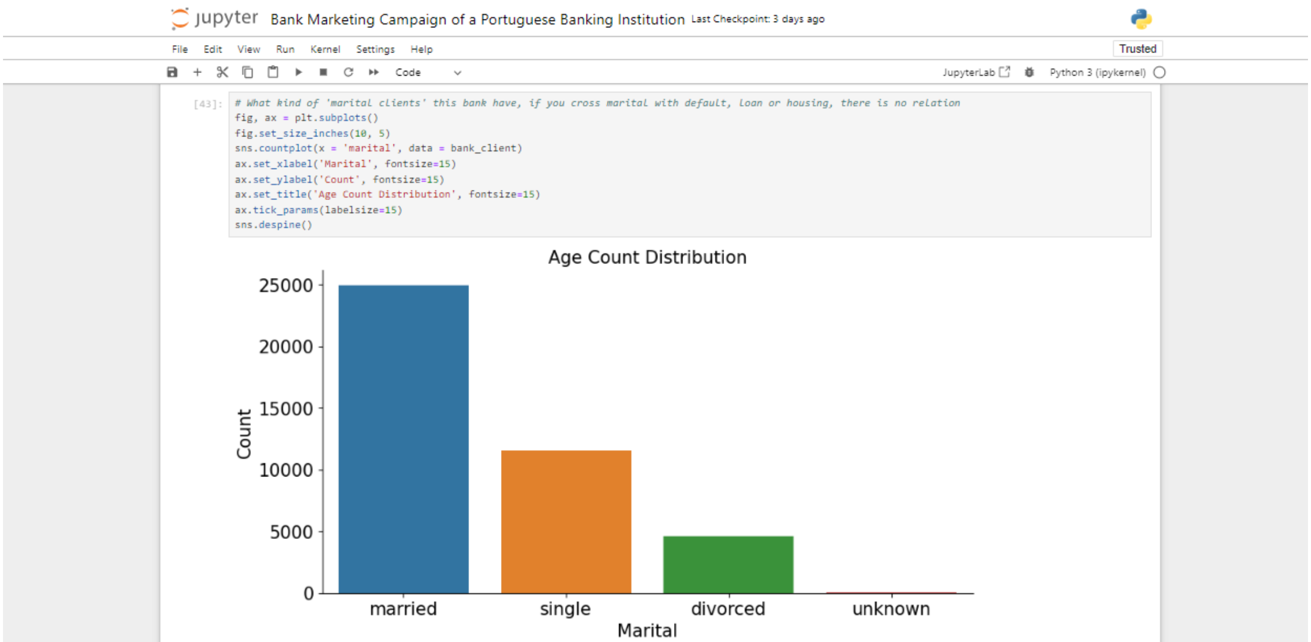


Figure 12. Marital count distribution



Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 13. Education count distribution

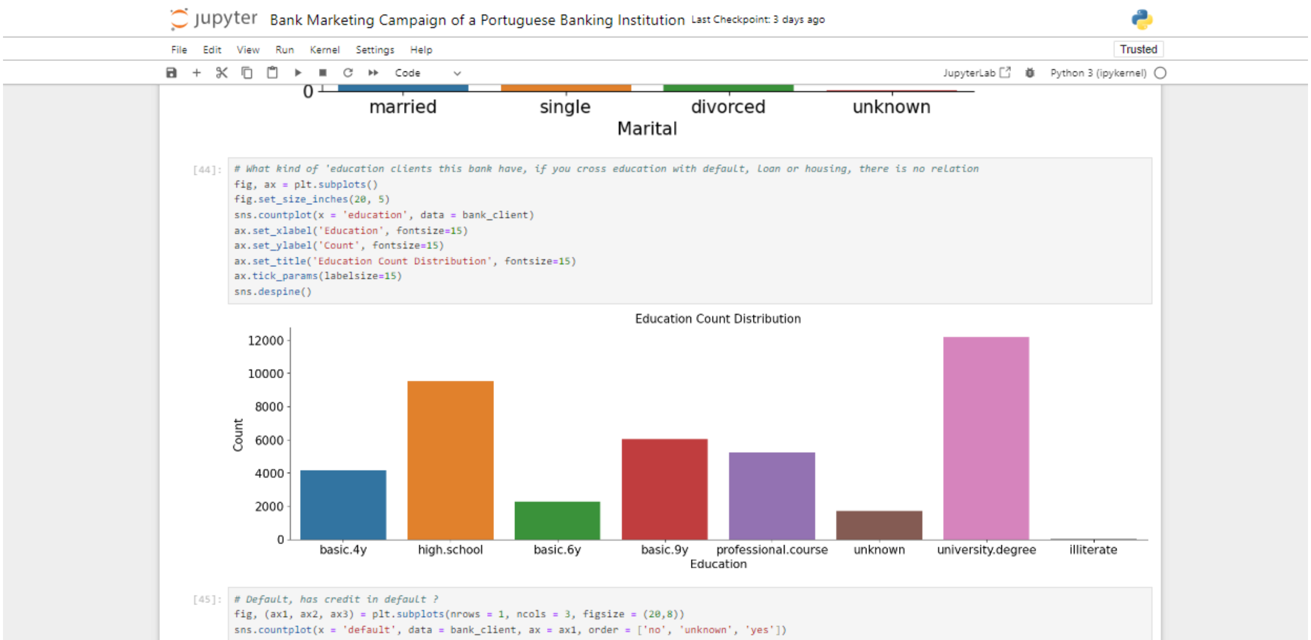
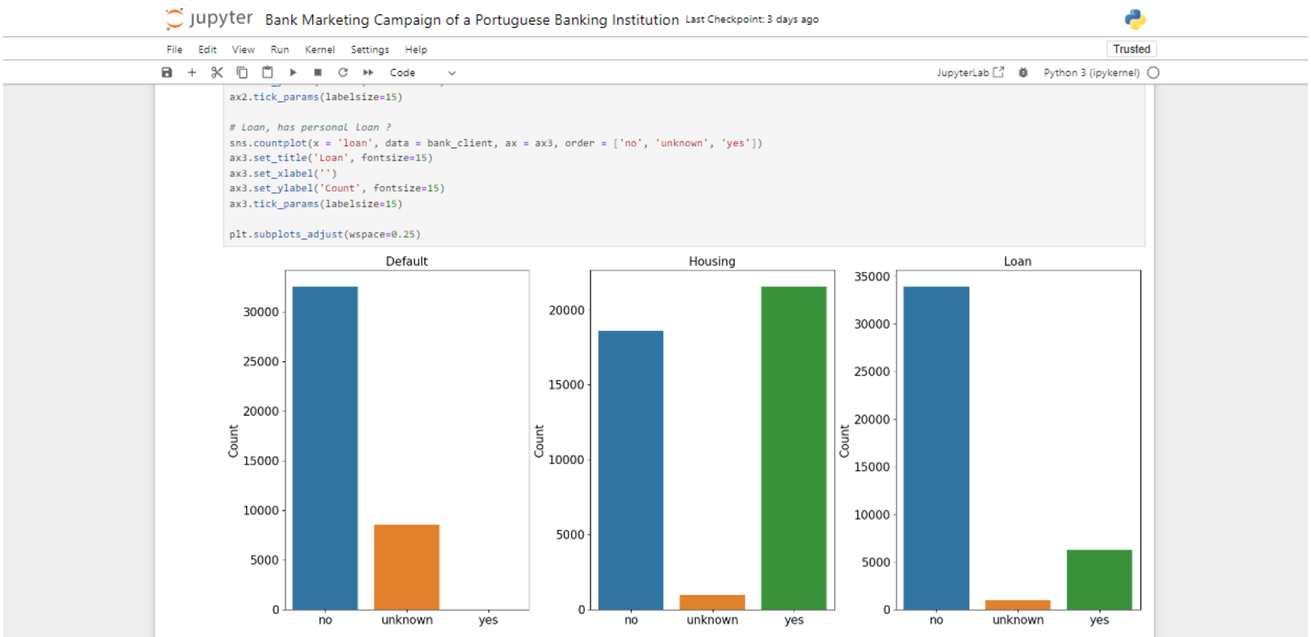
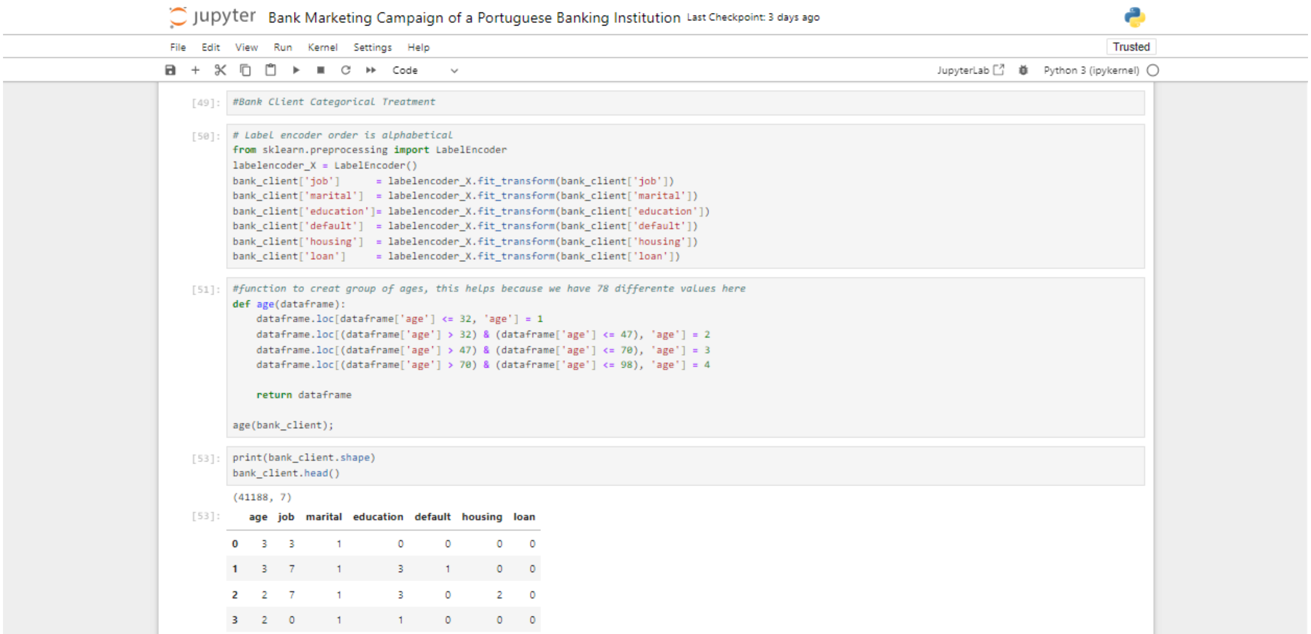


Figure 14. Default, housing, and loan distribution



Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 15. Bank client categorical treatment

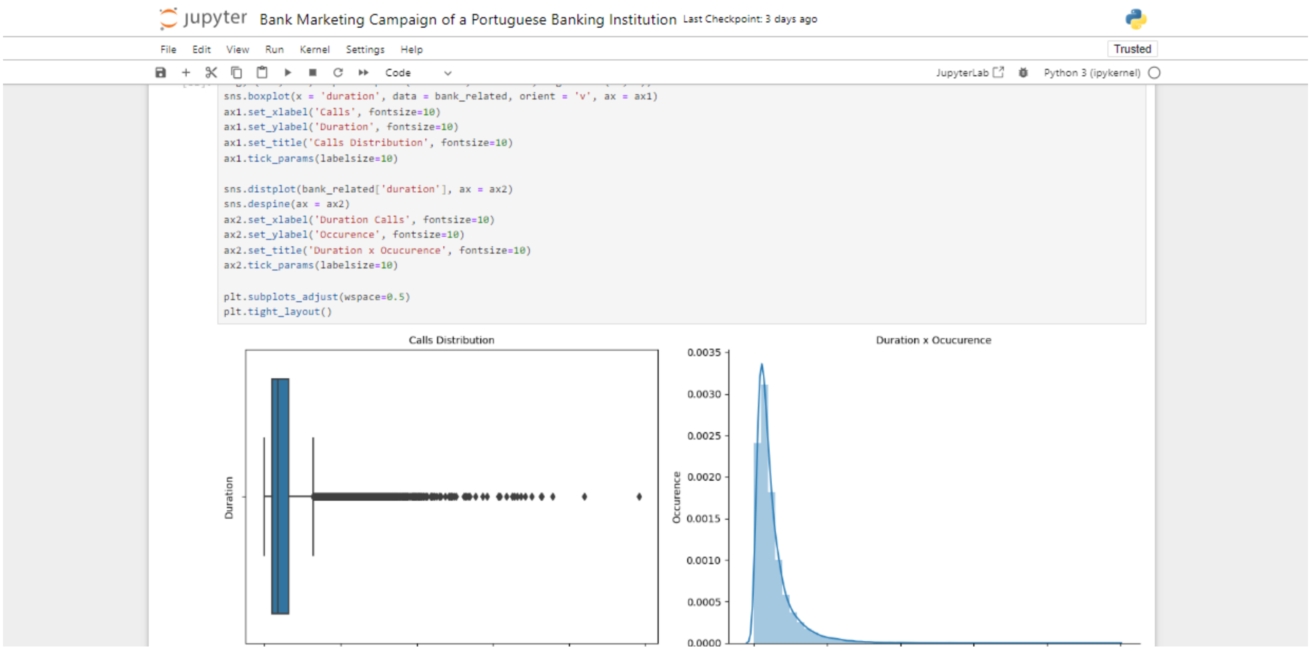


jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

```
[49]: #Bank Client Categorical Treatment

[50]: # Label encoder order is alphabetical
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_client['job']      = labelencoder_X.fit_transform(bank_client['job'])
bank_client['marital']  = labelencoder_X.fit_transform(bank_client['marital'])
bank_client['education']= labelencoder_X.fit_transform(bank_client['education'])
bank_client['default']  = labelencoder_X.fit_transform(bank_client['default'])
bank_client['housing']  = labelencoder_X.fit_transform(bank_client['housing'])
bank_client['loan']     = labelencoder_X.fit_transform(bank_client['loan'])

[51]: #function to creat group of ages, this helps because we have 78 differente values here
def age(dataframe):
    dataframe.loc[dataframe['age'] <= 32, 'age'] = 1
    dataframe.loc[(dataframe['age'] > 32) & (dataframe['age'] <= 47), 'age'] = 2
    dataframe.loc[(dataframe['age'] > 47) & (dataframe['age'] <= 70), 'age'] = 3
    dataframe.loc[(dataframe['age'] > 70) & (dataframe['age'] <= 98), 'age'] = 4

    return dataframe

age(bank_client);

[53]: print(bank_client.shape)
bank_client.head()

(41188, 7)
```

	age	job	marital	education	default	housing	loan
0	3	3	1	0	0	0	0
1	3	7	1	3	1	0	0
2	2	7	1	3	0	2	0
3	2	0	1	1	0	0	0

Figure 16. Calls distribution duration



jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

```
sns.boxplot(x = 'duration', data = bank_related, orient = 'v', ax = ax1)
ax1.set_xlabel('Calls', fontsize=10)
ax1.set_ylabel('Duration', fontsize=10)
ax1.set_title('Calls Distribution', fontsize=10)
ax1.tick_params(labelsize=10)

sns.distplot(bank_related['duration'], ax = ax2)
sns.despine(ax = ax2)
ax2.set_xlabel('Duration Calls', fontsize=10)
ax2.set_ylabel('Occurence', fontsize=10)
ax2.set_title('Duration x Ocucurence', fontsize=10)
ax2.tick_params(labelsize=10)

plt.subplots_adjust(wspace=0.5)
plt.tight_layout()
```


Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 17. Data pre-processing

jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

0.0 0.2 0.4 0.6 0.8 1.0 0.0 0.2 0.4 0.6 0.8 1.0 0.0 0.2 0.4 0.6 0.8 1.0

[75]:
```
print('Ages above: ', bank_related['duration'].quantile(q = 0.75) +
                      1.5*(bank_related['duration'].quantile(q = 0.75) - bank_related['duration'].quantile(q = 0.25)), 'are outliers')
```
```
Ages above:  644.5 are outliers
```

[76]:
```
bank_related[bank_related['duration'] > 640].count()
```

[76]:
```
contact        3008
month          3008
day_of_week    3008
duration       3008
dtype: int64
```

[77]:
```
# Label encoder order is alphabetical
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_related['contact']     = labelencoder_X.fit_transform(bank_related['contact'])
bank_related['month']       = labelencoder_X.fit_transform(bank_related['month'])
bank_related['day_of_week'] = labelencoder_X.fit_transform(bank_related['day_of_week'])
```

[78]:
```
bank_related.head()
```

[78]:

	contact	month	day_of_week	duration
0	1	6	1	261
1	1	6	1	149
2	1	6	1	226
3	1	6	1	151
4	1	6	1	307

[79]:
```
def duration(data):
```

Figure 18. Duration and attributes

jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
    data.loc[(data['duration'] > 180) & (data['duration'] <= 319)  , 'duration']   = 3
    data.loc[(data['duration'] > 319) & (data['duration'] <= 644.5), 'duration'] = 4
    data.loc[data['duration']  > 644.5, 'duration'] = 5

    return data
duration(bank_related);
```

[80]:
```
bank_related.head()
```

[80]:

	contact	month	day_of_week	duration
0	1	6	1	3
1	1	6	1	2
2	1	6	1	3
3	1	6	1	2
4	1	6	1	3

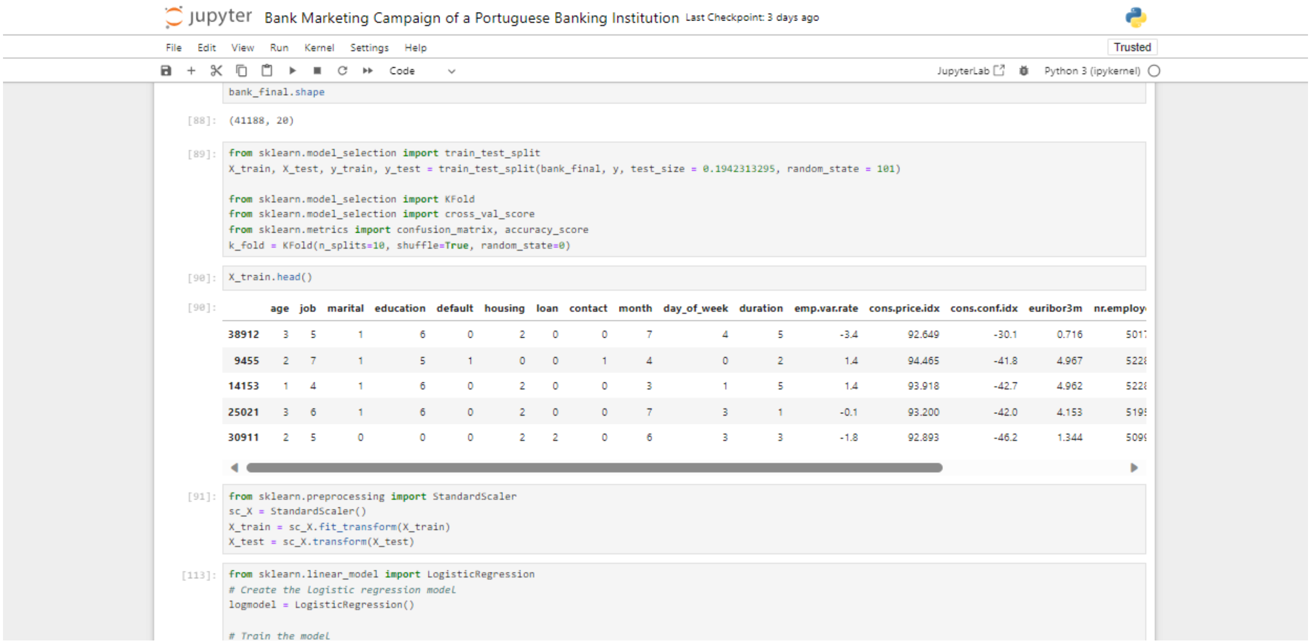
[81]:
```
#Social and economic context attributes
```

[82]:
```
bank_se = bank.loc[: , ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']]
bank_se.head()
```

[82]:

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
0	1.1	93.994	-36.4	4.857	5191.0
1	1.1	93.994	-36.4	4.857	5191.0
2	1.1	93.994	-36.4	4.857	5191.0
3	1.1	93.994	-36.4	4.857	5191.0
4	1.1	93.994	-36.4	4.857	5191.0

Figure 19. Train test split the data



jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
bank_final.shape
```

[88]: (41188, 20)

[89]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(bank_final, y, test_size = 0.1942313295, random_state = 101)

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

[90]:
```
X_train.head()
```

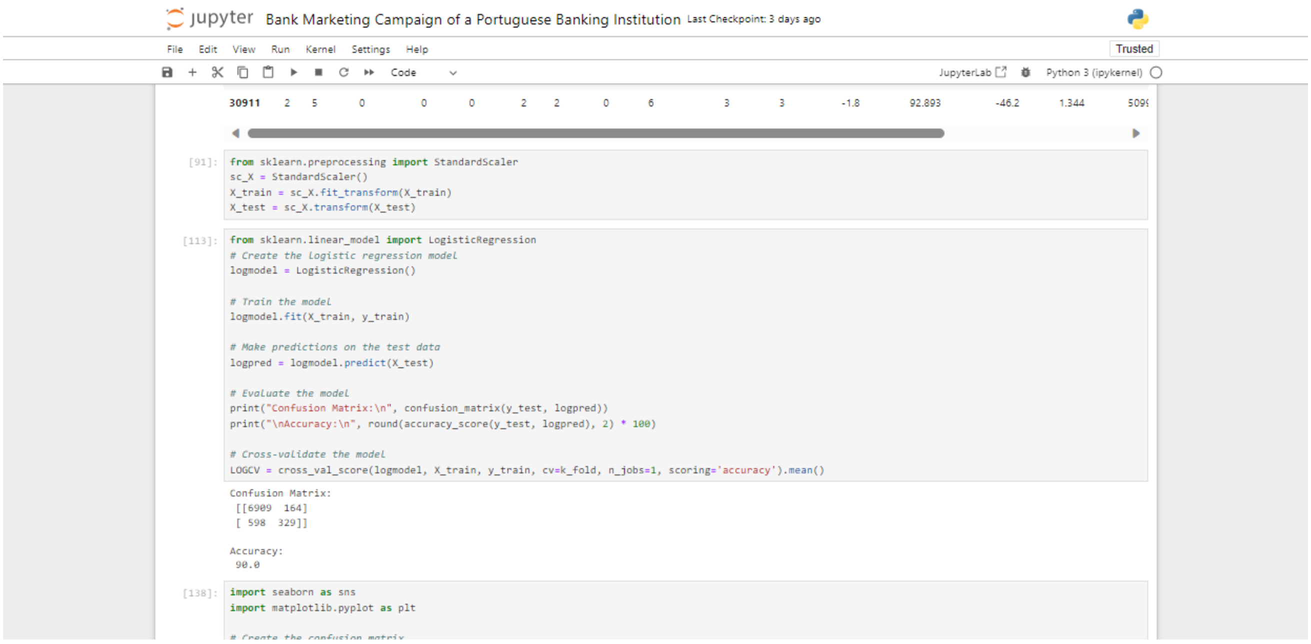
[90]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employ
38912	3	5	1	6	0	2	0	0	7	4	5	-3.4	92.649	-30.1	0.716	501
9455	2	7	1	5	1	0	0	1	4	0	2	1.4	94.465	-41.8	4.967	522
14153	1	4	1	6	0	2	0	0	3	1	5	1.4	93.918	-42.7	4.962	522
25021	3	6	1	6	0	2	0	0	7	3	1	-0.1	93.200	-42.0	4.153	519
30911	2	5	0	0	0	2	2	0	6	3	3	-1.8	92.893	-46.2	1.344	509

[91]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

[113]:
```
from sklearn.linear_model import LogisticRegression
# Create the logistic regression model
logmodel = LogisticRegression()

# Train the model
```

Figure 20. Logistic Regression Model



jupyter Bank Marketing Campaign of a Portuguese Banking Institution Last Checkpoint: 3 days ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

30911	2	5	0	0	0	2	2	0	6	3	3	-1.8	92.893	-46.2	1.344	509

[91]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

[113]:
```
from sklearn.linear_model import LogisticRegression
# Create the logistic regression model
logmodel = LogisticRegression()

# Train the model
logmodel.fit(X_train, y_train)

# Make predictions on the test data
logpred = logmodel.predict(X_test)

# Evaluate the model
print("Confusion Matrix:\n", confusion_matrix(y_test, logpred))
print("\nAccuracy:\n", round(accuracy_score(y_test, logpred), 2) * 100)

# Cross-validate the model
LOGCV = cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean()
```

```
Confusion Matrix:
 [[6909  164]
 [ 598  329]]

Accuracy:
 90.0
```

[138]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the confusion matrix
```

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

Figure 21. Confusion Matrix for Logistic Regression

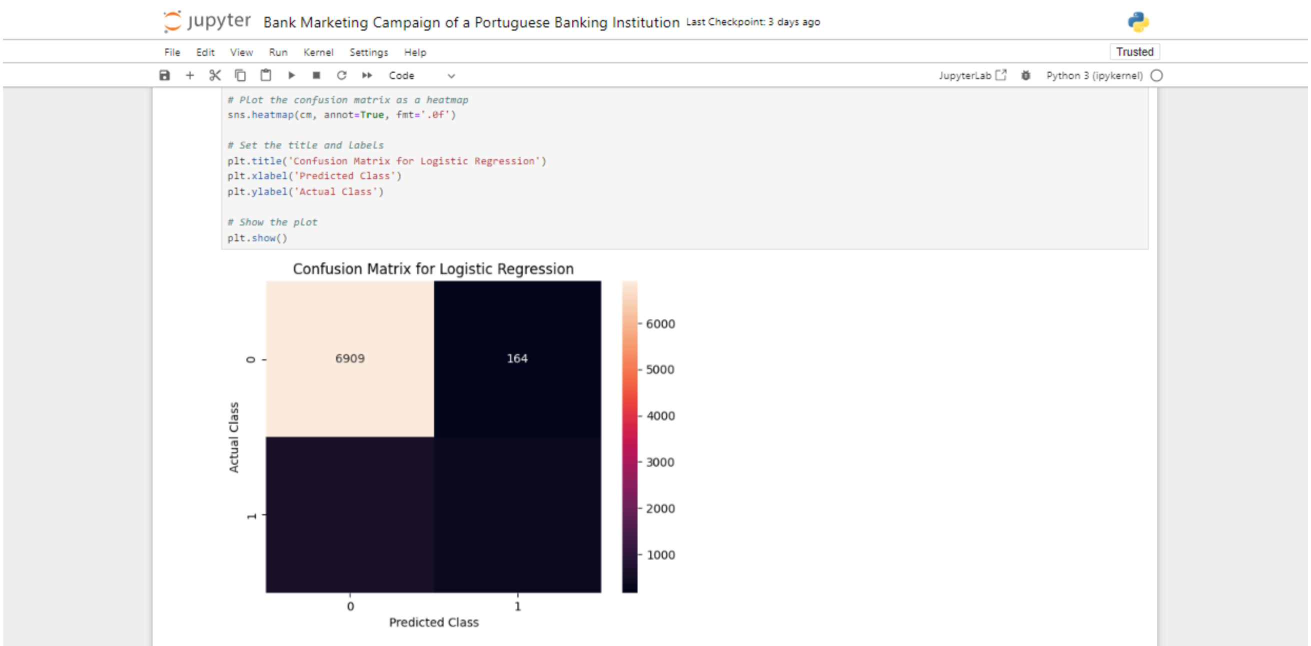


Figure 22. ROC Curve for Logistic Regression

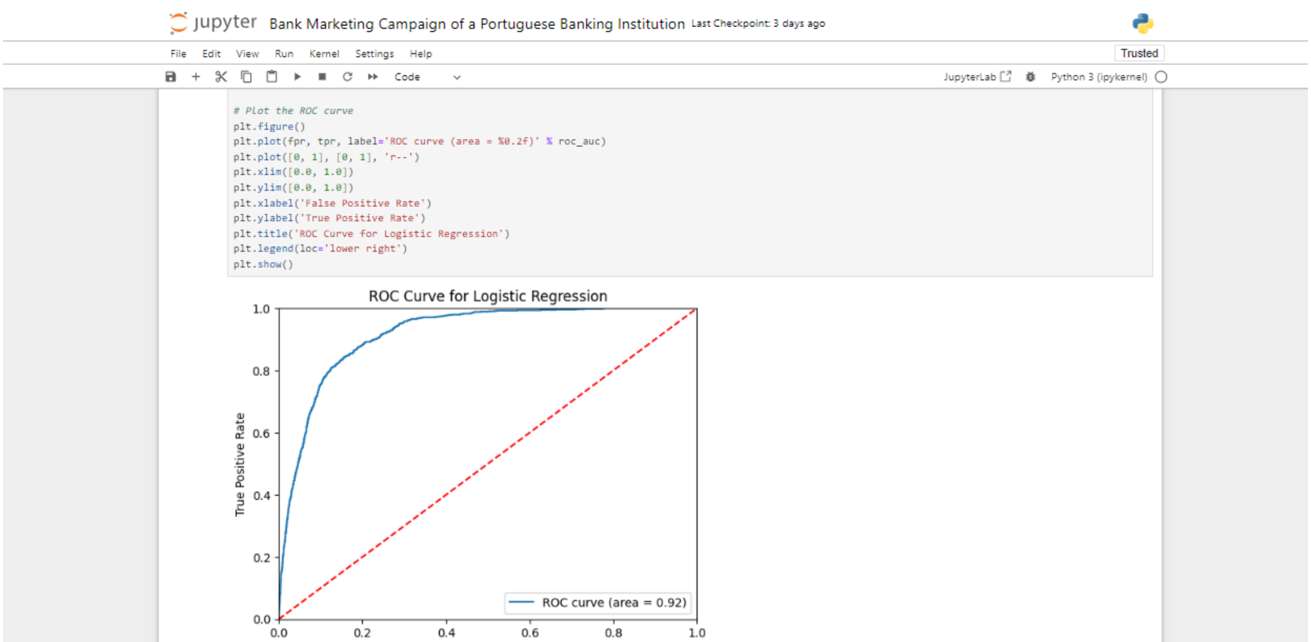


Figure 23. Metrics Classification Report

```
[108]: from sklearn.metrics import classification_report

       # Calculate the classification report for the Logistic Regression model
       report = classification_report(y_test, logpred)

       # Print the classification report
       print(report)
```

```
              precision    recall  f1-score   support

       False       0.92      0.98      0.95      7073
        True       0.67      0.35      0.46       927

    accuracy                           0.90      8000
   macro avg       0.79      0.67      0.71      8000
weighted avg       0.89      0.90      0.89      8000
```

```
[ ]:
```