CONTENTS

1. INTRODUCTION

- 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
- The classification goal is to predict if the client will subscribe a term deposit (variable y) or not.

This problem can be addressed as a **binary classification** problem.

UCI repository link is given below.

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

2. DATA

There were 16 features and 1 output variable can be seen in the dataset.

```
[6] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

```
data.shape

(45211, 17)
```

```
data.columns

Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

```
data.describe(include='all').transpose()
```

1 to 17 of 17 entries Filter

index	count	unique	top	freq	mean	std	min	25%	50%	75%	max
age	45211.0	NaN	NaN	NaN	40.93621021432837	10.61876204097542	18.0	33.0	39.0	48.0	95.0
job	45211	12	blue-collar	9732	NaN	NaN	NaN	NaN	NaN	NaN	NaN
marital	45211	3	married	27214	NaN	NaN	NaN	NaN	NaN	NaN	NaN
education	45211	4	secondary	23202	NaN	NaN	NaN	NaN	NaN	NaN	NaN
default	45211	2	no	44396	NaN	NaN	NaN	NaN	NaN	NaN	NaN
balance	45211.0	NaN	NaN	NaN	1362.2720576850766	3044.765829168518	-8019.0	72.0	448.0	1428.0	102127.0
housing	45211	2	yes	25130	NaN	NaN	NaN	NaN	NaN	NaN	NaN
loan	45211	2	no	37967	NaN	NaN	NaN	NaN	NaN	NaN	NaN
contact	45211	3	cellular	29285	NaN	NaN	NaN	NaN	NaN	NaN	NaN
day	45211.0	NaN	NaN	NaN	15.80641879188693	8.322476153044592	1.0	8.0	16.0	21.0	31.0
month	45211	12	may	13766	NaN	NaN	NaN	NaN	NaN	NaN	NaN
duration	45211.0	NaN	NaN	NaN	258.1630797814691	257.5278122651719	0.0	103.0	180.0	319.0	4918.0
campaign	45211.0	NaN	NaN	NaN	2.763840658246887	3.0980208832791694	1.0	1.0	2.0	3.0	63.0
pdays	45211.0	NaN	NaN	NaN	40.19782796222158	100.12874599059835	-1.0	-1.0	-1.0	-1.0	871.0
previous	45211.0	NaN	NaN	NaN	0.5803233726305546	2.3034410449312213	0.0	0.0	0.0	0.0	275.0
poutcome	45211	4	unknown	36959	NaN	NaN	NaN	NaN	NaN	NaN	NaN
y	45211	2	no	39922	NaN	NaN	NaN	NaN	NaN	NaN	NaN

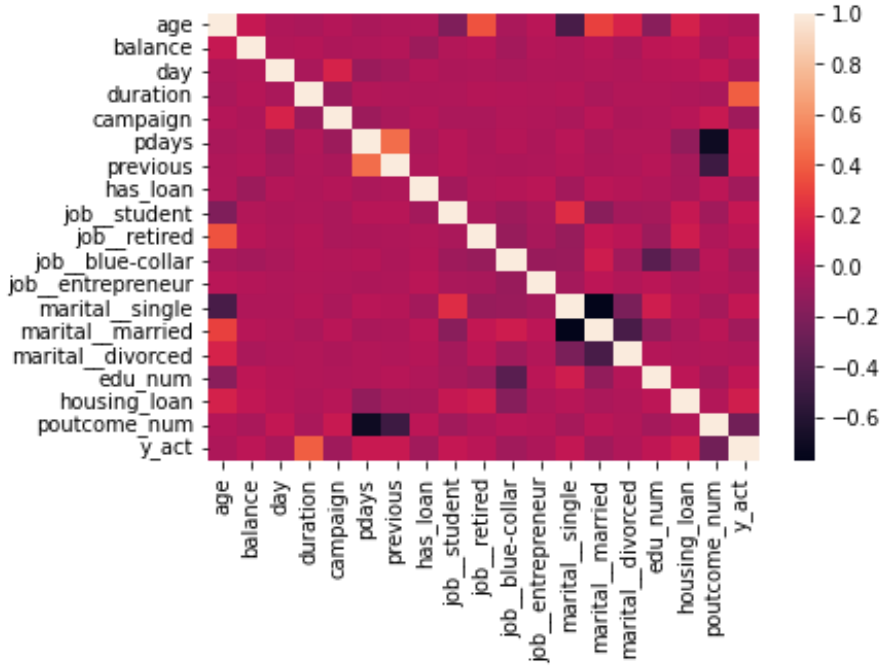
3. METHODOLOGY

The following mentioned procedure was adopted to design the final ML solution.

- Load Data to the Colab environment
  - *'https://raw.githubusercontent.com/ajithdolamulla/Capstone-Project/main/bank-full.csv'*
- Inspecting loaded Dataset
  - *Ex. data.info(),data.shape, data.describe(include='all').transpose(),data['y'].unique() etc.*

- Pre-Process Data for Training
  - Transform into Binary Column – ex. *data['y_act']=np.where(data['y'] == 'no',0,1) ,*
  - Create ID Column – ex. *data['id'] = data.index+1*
  - Rearranging Columns
  - Descriptive stats and identify Ranges – ex. *data[' '].describe function*
  - Outlier treatment and Plot distributions – ex. *data['age'].plot(kind='box') , find Outliers using Percentiles.*
  - working on categorical columns – ex. *One hot encoding (1/0, create dummy variables)*
  - Label Encoding (ordinal) for education categorical column
  -
- Data Pre-processing Function – *Pre-process the input parameters and select features for the ML model*
- Train Test Split – *train_test_split function in Scikit-learn*
- Manually explore hyperparameter space- *Random Forrest and Logistic Regression algorithm*
- Use of Grid Search – *GridSearchCV in Scikit-learn*
- Select Best Model – *select the best model considering accuracy, precision, F1Score, roc_auc*
- Saving Best Model – *using pickle or joblib*
- Score Function - *model.predict_proba(input_data)*
- Post-processing Function for Prediction
- Prediction Function for Application (Inference Pipeline) – *Test with sample data*
- Local Web API development – using POSTman tool and validated by jupyter notebook.
- Feature Importance *using LIME and SHAP models to explain the feature importance*

4. RESULTS




```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print(F"Train sample size = {len(X_train)}")
print(F"Test sample size  = {len(X_test)}")
```

```
Train sample size = 31148
Test sample size  = 13350
```

```
[ ]  model0
```

```
{'model_name': 'rf_new',
 'model': RandomForestClassifier(max_depth=10, n_estimators=500, n_jobs=3, verbose=1),
 'accuracy': 0.9068913857677903,
 'precision': 0.6881533101045296,
 'f1_score': 0.8882957625888304,
 'roc_auc': 0.8941356111709002}
```

	model_name	model	accuracy	precision	f1_score	roc_auc
0	lgr1	LogisticRegression(n_jobs=3, verbose=1)	0.900974	0.622540	0.880697	0.852467
1	rf1	(DecisionTreeClassifier(max_features='auto', r...	0.906816	0.629674	0.895190	0.886337
2	rf2	(DecisionTreeClassifier(max_features='auto', r...	0.906142	0.620609	0.895032	0.890946
3	rf3	(DecisionTreeClassifier(max_depth=10, max_feat...	0.906966	0.690685	0.888230	0.893846
4	rf4	(DecisionTreeClassifier(max_depth=20, max_feat...	0.906966	0.630566	0.895406	0.891486

```
[ ] from sklearn.model_selection import GridSearchCV

    parameters = {'n_estimators': [100,500], 'max_depth': [None, 10, 20]}
    gs_model = GridSearchCV(RandomForestClassifier(), parameters, n_jobs=2, verbose=3, pre_dispatch=2)
    gs_model.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/joblib/externals/loky/process_executor.py:703: UserWa
  "timeout or by a memory leak.", UserWarning
GridSearchCV(estimator=RandomForestClassifier(), n_jobs=2,
             param_grid={'max_depth': [None, 10, 20],
                         'n_estimators': [100, 500]},
             pre_dispatch=2, verbose=3)
```

```
print(gs_model.best_params_)
```

```
{'max_depth': 20, 'n_estimators': 500}
```

```
[ ] # Select best model
    model = models.query("model_name=='rf3'")
    model
```

	model_name	model	accuracy	precision	f1_score	roc_auc
3	rf3	(DecisionTreeClassifier(max_depth=10, max_feat...	0.906966	0.690685	0.88823	0.893846

```
[ ] model = model['model'].values[0]
    model
```

```
RandomForestClassifier(max_depth=10, n_estimators=500, n_jobs=3, verbose=1)
```

5. CONCLUSION

This problem was treated as a binary classification problem. First the data set was analysed and prepare for the model training. Then the Data Pre-processing task was executed to make a reusable code for the efficiency of the model deployment. For the best ML algorithm was selected using hyperparameter tuning process. For the selection of ML algorithm Logistic regressing and random forest algorithms were used. For the selection criteria accuracy, Precision, Recall and F1 score parameters were considered. Once Random Forest Algorithm was selected, it was further fine tune using Grid search function. Then the best model with parameters were selected.

Further the inference pipeline was developed using Locally hosted Web API.

6. DISCUSSION

I have selected a data set in UCI repository https://archive.ics.uci.edu/ml/datasets/Bank+Marketing which is relevant to bank marketing.

As the first step this problem was analysed based on the Input variables and the output variable. Based on the requirement of the bank this case can be categorized under a classification type problem with supervised learning algorithm.

The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y) so this problem falls into binary classification problem.

After loading the data set it was analyzed and prepare to design ML solution. Basically, this covers

- Transform into Binary Column
- Create ID columns
- One hot encoding (1/0, create dummy variables)
- Label Encoding (ordinal) for education categorical column
- Outlier treatment and Plot distributions
- Correlation plots
- Select columns etc.

```
def pre_processing(data):
    data['id'] = data.index+1

    data['has_loan']=np.where(data['loan'] == 'no',0,1)
    data['housing_loan']=np.where(data['housing'] == 'no',1,0)
    data['job_grp'] = data['job']
    data = data.join(pd.get_dummies(data['job_grp'], prefix='job_'))
    data['marital_grp'] = data['marital']
    data = data.join(pd.get_dummies(data['marital_grp'], prefix='marital_'))

    data['edu_num'] = data['education']
    data['edu_num'].replace('primary', 1, inplace=True)
    data['edu_num'].replace('secondary', 2, inplace=True)
    data['edu_num'].replace('tertiary', 3, inplace=True)
    data['edu_num'].replace('unknown', 0, inplace=True)

    data['poutcome_num'] = data['poutcome']
    data['poutcome_num'].replace('failure', 3, inplace=True)
    data['poutcome_num'].replace('other', 2, inplace=True)
    data['poutcome_num'].replace('success', 1, inplace=True)
    data['poutcome_num'].replace('unknown', 4, inplace=True)



    # Select Columns
    X_variables = ['age', 'balance', 'day','duration', 'campaign', 'pdays','previous','has_loan','job__student',
                   'job__retired','job__blue-collar','job__entrepreneur','marital__single','marital__married','marital__divorced',
                   'edu_num','housing_loan','poutcome_num']
```

Then the working data set was split in to Training and Testing (70:30) using **train_test_split** function in **Scikit-learn.**

Based on the model training below output was extracted.

{'model_name': 'rf_new',

'model': RandomForestClassifier(max_depth=10, n_estimators=500, n_jobs=3, verbose=1),

'accuracy': 0.9067415730337078,

'precision': 0.6870629370629371,

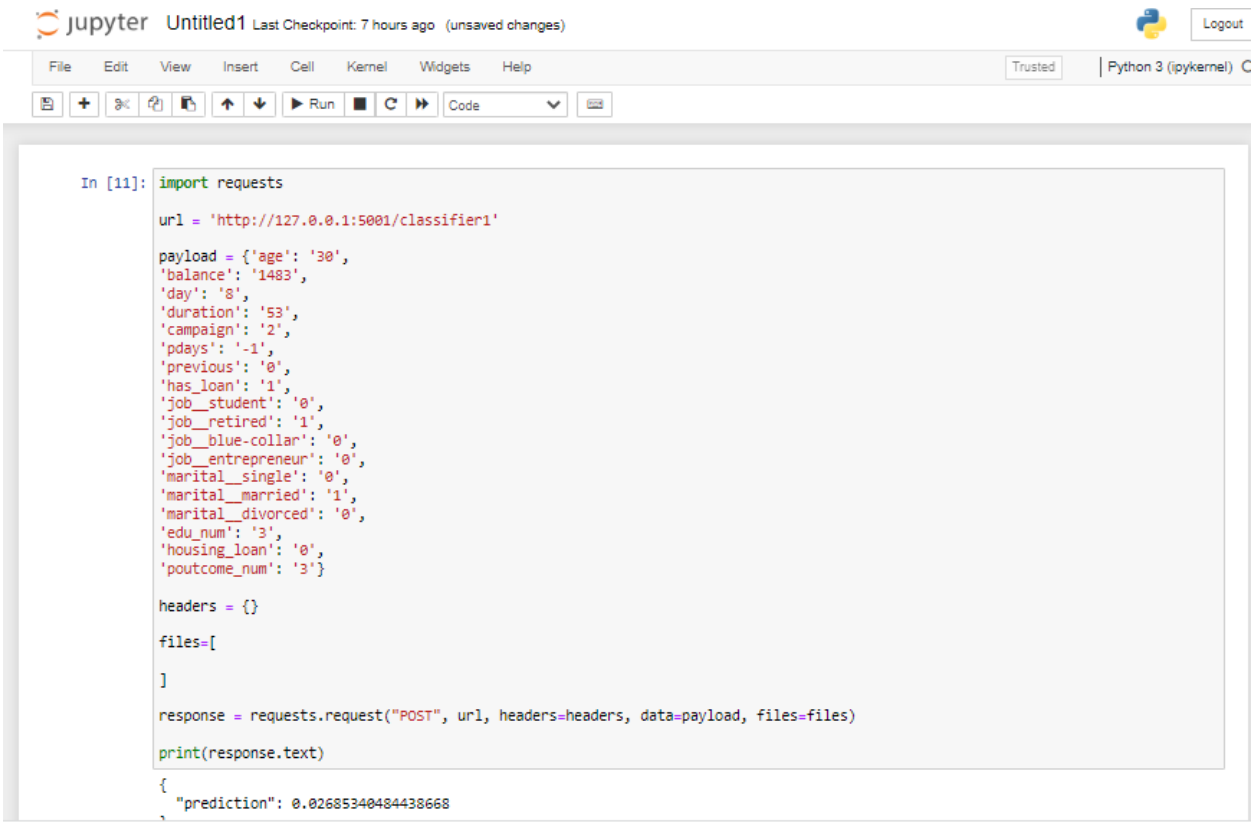
'f1_score': 0.888053767890051,

'roc_auc': 0.8943157947887278}

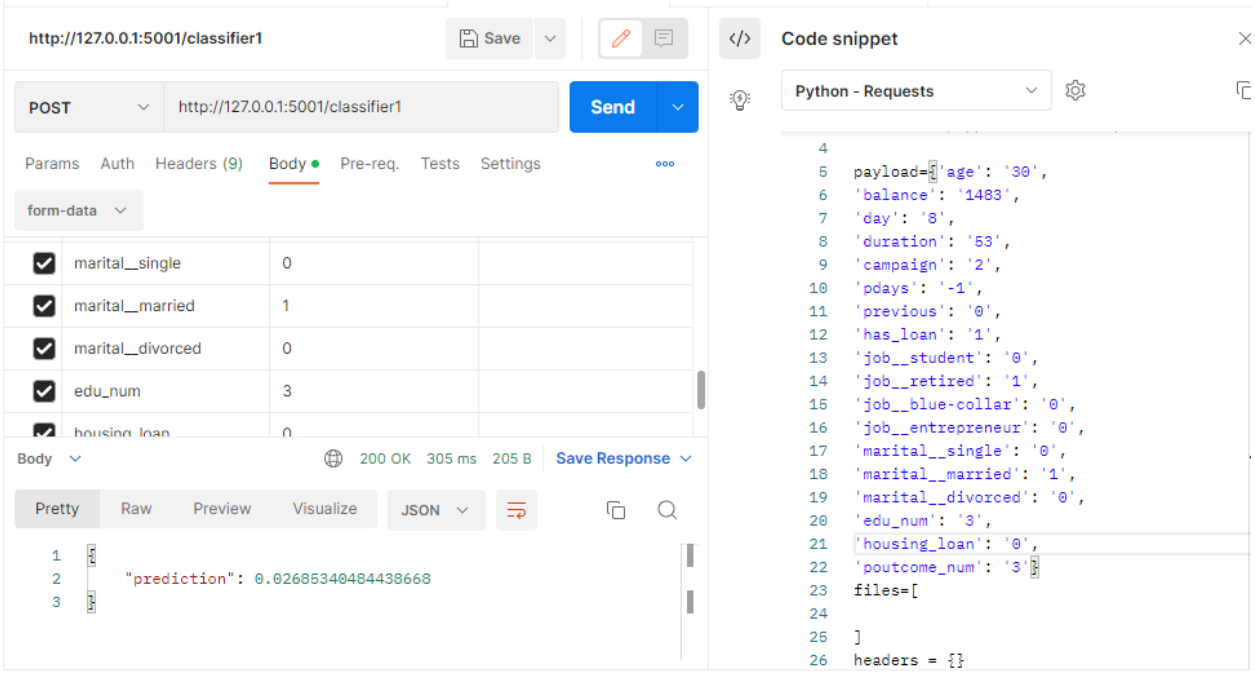
The best model was found through hyperparameter tuning and Grid search methods.

Then the feature importance was analysed by LIME and SHAP models.

<u>Locally hosted Web API – using jupyter notebook</u>



Using POSTman tool



Application development Using Streamlit