Import Packages

In [75]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline


from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import keras
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense,Dropout
from imblearn.over_sampling import SMOTE
from statistics import mode

import warnings
warnings.filterwarnings('ignore')
```

Feature Builder Utility

In [52]:
```
def feature_builder(inp_data):
  # Remove Numeric Features outliers based on the EDA
  inp_data["age"] = np.where(inp_data["age"] > 71, 71, inp_data['age'])
  inp_data["duration"] = np.where(inp_data["duration"] > 1260, 1260, inp_data['duratio
  inp_data["campaign"] = np.where(inp_data["campaign"] > 14, 14, inp_data['campaign'])
  inp_data["previous"] = np.where(inp_data["previous"] > 3, 3, inp_data['previous'])
  # Column Transformers
  # Scaling for Numeric Columns
  # One Hot Encoding for Categoriccal Features
  numerical_features = ['age','duration', 'campaign', 'previous']
  categorical_features = [ 'job', 'marital', 'education', 'default', 'housing', 'loan
  feature_transformer = make_column_transformer((StandardScaler(), numerical_features)
  # Create Feature Data
  feature_array = feature_transformer.fit_transform(inp_data)
  feature_cols = list(feature_transformer.get_feature_names_out())
  feature_cols = [col.replace('standardscaler__', '').replace('onehotencoder__', '').r
  feature_data = pd.DataFrame.sparse.from_spmatrix(feature_array,columns=feature_cols)
  return feature_transformer,feature_data
```

Model Training

In [55]:
```
train_data = pd.read_excel('https://github.com/rsdevanathan/Customer_Subscription/blob
```

In [56]:
```
# Remove Duplicates
train_data = train_data.drop_duplicates()
#Encode Target Variable
train_data = train_data.replace({'y': {'yes': 1,
                                       'no': 0}})
```

In [57]:
```
X_train = train_data.drop(columns = 'y')
y_train = train_data[['y']]
cw = {0:1,1:np.count_nonzero(y_train==0)/np.count_nonzero(y_train==1)}
```

In [58]:
```
feature_transformer_pipeline,X_train = feature_builder(X_train)
```

In [59]:
```
smote_sampler = SMOTE(random_state=14)
columns = X_train.columns
arr_X, arr_y = smote_sampler.fit_resample(X_train.values, y_train)
sampled_X = pd.DataFrame(data=arr_X,columns=X_train.columns )
sampled_y = pd.DataFrame(data=arr_y,columns=['y'])
```

Logistic Regression

In [60]:
```
ml_lr = LogisticRegression(max_iter=1000)
ml_lr.fit(sampled_X,sampled_y)
kfold = StratifiedKFold(n_splits=3, shuffle=True)
cv_auc = cross_val_score(ml_lr, X_train.values, y_train.values, scoring='roc_auc',cv=
print("Training AUC:",np.mean(cv_auc))
```

```
Training AUC: 0.9126389285216318
```

Random Forest

In [74]:
```
ml_rf=RandomForestClassifier(random_state=14,class_weight = cw,max_depth=100,max_featu
ml_rf.fit(X_train, y_train)
kfold = StratifiedKFold(n_splits=3, shuffle=True)
cv_auc = cross_val_score(ml_rf, X_train.values, y_train.values, scoring='roc_auc',cv=
print("Training AUC:",np.mean(cv_auc))
```

```
Training AUC: 0.9266264073790428
```

Neural Network

In [76]:
```
def create_nn():
  model = Sequential()
  model.add(Dense(32, input_dim=57, activation='relu'))
  model.add(Dense(16, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['binary_accura
  return model
dl_cfr_cw = KerasClassifier(build_fn=create_nn, epochs=100, batch_size=1028, verbose=
dl_cfr_cw.fit(X_train, y_train,class_weight = cw)
kfold = StratifiedKFold(n_splits=3, shuffle=True)
cv_auc = cross_val_score(dl_cfr_cw, X_train.values, y_train.values, scoring='roc_auc',
print("Training AUC:",np.mean(cv_auc))
```

```
Training AUC: 0.9211359915936056
```

Model Prediction

In [61]:
```
# Read Input Files
test_data = pd.read_excel('https://github.com/rsdevanathan/Customer_Subscription/blob/
```

In [62]:
```
test_data["age"] = np.where(test_data["age"] > 71, 71, test_data['age'])
test_data["duration"] = np.where(test_data["duration"] > 1260, 1260, test_data['durati
test_data["campaign"] = np.where(test_data["campaign"] > 14, 14, test_data['campaign']
test_data["previous"] = np.where(test_data["previous"] > 3, 3, test_data['previous'])
```

In [63]:
```
test_features = feature_transformer_pipeline.transform(test_data)
```

In [78]:
```
lr_pred = ml_lr.predict(test_features)
rf_pred = ml_rf.predict(test_features)
xgb_pred = dl_cfr_cw.predict(test_features)
```

In [79]:
```
prediction_data = test_data.copy()
prediction_data['LR_Prediction'] = lr_pred
prediction_data['RF_Prediction'] = rf_pred
prediction_data['XGB_Prediction'] = xgb_pred
```

In [86]:
```
prediction_data['Final_Prediction'] = np.where((prediction_data['LR_Prediction']+predi
```

In [90]:
```
prediction_data['Final_Prediction'].value_counts()
```

Out[90]:
```
0    34
1     6
Name: Final_Prediction, dtype: int64
```

In [91]:
```
prediction_data.to_csv('Final_Prediction.csv')
```

End