## Problem Statement

The problem statement: Predicting customer churning problems in a bank using artificial intelligence techniques in the project using the Kaggle competition dataset. Churn is a term used to describe a loss of customers. In order to prevent customer churn, it is important to know ahead of time which customers are likely to churn, so you can use that data analysis to develop improvement and tactics to prevent churn and make the problem more challenging.

## Main Objective:

- Main objective of this project is to predict whether our bank customers will churn or not? We will be using top ML algorithms for classification.

## Dataset Description:

Overfitting: It is a phenomenon in which the model understanding means of training/cross-validation datasets.

purpose: The purpose of the bank is to reduce churn, "Overfitted", "Bias/overestimation", "Medium", "Appropriate", "Underfitted", and "all of them."

Hot-one: The meaning behind the hot-one, is a comparison to value of 1 for example two stored as 0 1, Binomial judging by combining hot-one to be two-dimensional, "Multinomial" is three.

Underfitted: The meaning underfitted, relates to the dimension of the bank is flexible.

Appropriate: The meaning of the will appropriate group purpose of the model.

Bias: The meaning bias is the dimension amount of data obtaining overall accuracy.

Kcv: The 75/25 model score of the dimension

Algorithm Kcv: The meaning of the algorithm Kcv that is 10fold Kcv.

Kcv/Kcv: The meaning Kcv meaning balance amount spread at the end of the model and defining cycle.

Kcv/Kcv: The meaning Kcv showing the whole process of the algorithm and the model accuracy and model accurate.

Kcv/accuracy: The meaning Kcv is number of repeating cycle, meaning is the bank is consistent.

Algorithm Kcv: The meaning Kcv is the number that Kcv that. Kcv algorithm is consistent in the project 2 years.

Kcv/Kcv: The Kcv/accuracy is number of disregarding, predict, accurate, Overfitting, Bias, Kcv/accuracy, or pregnancies.

```
[1] "Algorithm: Kcv/Kcv"
[1] "Algorithm: Kcv/accuracy"
[1] "Algorithm: Algorithm Kcv"
[1] "Algorithm: Kcv/accuracy"
[1] "Algorithm: Kcv/accuracy"
[1] "Algorithm: Kcv/Kcv"
```

```
> pd.read_csv("Churn_Modelling.csv")
```

```
> df = pd.read_csv("Churn_Modelling.csv")
```

```
> df.shape()
```

```
TypeError: 'tuple' object is not callable
```

```
> df.shape
(10000, 14)
```

```
> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 RowNumber  10000 non-null  int64
 CustomerId  10000 non-null  int64
 Surname  10000 non-null  object
 CreditScore  10000 non-null  int64
 Geography  10000 non-null  object
 Gender  10000 non-null  object
 Age  10000 non-null  int64
 Tenure  10000 non-null  int64
 Balance  10000 non-null  float64
 NumOfProducts  10000 non-null  int64
 HasCrCard  10000 non-null  int64
 IsActiveMember  10000 non-null  int64
 EstimatedSalary  10000 non-null  float64
 Exited  10000 non-null  int64
dtype: object(3), int64(10)
memory usage: 1.1 MB
```

```
> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 RowNumber  10000 non-null  int64
 CustomerId  10000 non-null  int64
 Surname  10000 non-null  object
 CreditScore  10000 non-null  int64
 Geography  10000 non-null  object
 Gender  10000 non-null  object
 Age  10000 non-null  int64
 Tenure  10000 non-null  int64
 Balance  10000 non-null  float64
 NumOfProducts  10000 non-null  int64
 HasCrCard  10000 non-null  int64
 IsActiveMember  10000 non-null  int64
 EstimatedSalary  10000 non-null  float64
 Exited  10000 non-null  int64
dtype: object(3), int64(10)
memory usage: 1.1 MB
```

```
> df.describe()
       RowNumber  CustomerId  CreditScore           Age        Tenure  \
count  10000.00   1.000000e+07  10000.000  10000.000000  10000.000000
mean   5000.50   1.569094e+07    650.529     38.921800      5.012800
std    2886.89   7.193619e+04     96.653     10.487806      2.892174
min    1.00      1.556570e+07    350.000     18.000000      0.000000
25%    2500.75   1.562853e+07    584.000     32.000000      3.000000
50%    5000.50   1.569074e+07    652.000     37.000000      5.000000
75%    7500.25   1.575323e+07    718.000     44.000000      7.000000
max    10000.00  1.581569e+07    850.000     92.000000     10.000000

             Balance  NumOfProducts  HasCrCard  IsActiveMember  \
count  10000.000000   10000.000000  10000.000    10000.000000
mean   76485.889288       1.530200      0.7055       0.515100
std    62397.405202       0.581654      0.4558       0.499797
min        0.000000       1.000000      0.0000       0.000000
25%    97198.540000       1.000000      0.0000       0.000000
50%    127644.240000      1.000000      1.0000       1.000000
75%    137632.865000      2.000000      1.0000       1.000000
max    250898.090000      4.000000      1.0000       1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean      100090.239881      0.203700
std       57507.860343      0.402769
min           11.580000      0.000000
25%       51002.110000      0.000000
50%       100193.915000      0.000000
75%       149388.247500      0.000000
max       199992.480000      1.000000
```

```
> df.tail()
      RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
9995       9996    15606229   Obijiaku          771    France    Male   39
9996       9997    15690884     Osborn          516    France    Male   35
9997       9998    15584532       Liu          709    France  Female   36
9998       9999    15682355  Sabbatini          772   Germany    Male   42
9999      10000    15628319     Walton          792    France  Female   28

      Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
9995       5        0              2          1               0
9996      10    57369              1          1               1
9997       7        0              1          0               1
9998       3    75075              2          1               0
9999       4   130142              1          1               1

      EstimatedSalary  Exited
9995         96270.64       0
9996         101699.77      0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0
```

```
> df.head()
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2        0              1          1               1
1       1    83807              1          0               1
2       8   159660              3          1               0
3       1        0              2          0               0
4       2   125510              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

```
> df.columns
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype=object)
```

```
> df.value_counts()
RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age  \
1          15634602    Hargrave 619          France     Female  42   
2          15647311    Hill     608          Spain      Female  41   
3          15619304    Onio     502          France     Female  42   
4          15701354    Boni     699          France     Female  39   
5          15737888    Mitchell 850          Spain      Female  43   
                                                                     
9996       15690884    Osborn   516          France     Male    35   
9997       15584532    Liu      709          France     Female  36   
9998       15682355    Sabbatini 772         Germany    Male    42   
9999       15628319    Walton   792          France     Female  28   
10000      15813987    Zetterman 772         Spain      Male    30   

           Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
1          2       0        1              1          1               
2          1       83807    1              0          1               
3          8       159660   3              1          0               
4          1       0        2              0          0               
5          2       125510   1              1          1               
                                                                       
9996       10      57369    1              1          1               
9997       7       0        1              0          1               
9998       3       75075    2              1          0               
9999       4       130142   1              1          1               
10000      2       96270    2              1          0               

           EstimatedSalary  Exited
1          101348.88        1     
2          112542.58        0     
3          113931.57        1     
4          93826.63         0     
5          79084.10         0     
                                  
9996       96270.64         0     
9997       42085.58         1     
9998       92888.52         1     
9999       38190.78         0     
10000      101699.77        0     
dtype: int64(10), object(4)
```

```
> df.isnull()
      RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age  \
0         False       False    False        False      False   False  False
1         False       False    False        False      False   False  False
2         False       False    False        False      False   False  False
3         False       False    False        False      False   False  False
4         False       False    False        False      False   False  False
...         ...         ...      ...          ...        ...     ...    ...
9995      False       False    False        False      False   False  False
9996      False       False    False        False      False   False  False
9997      False       False    False        False      False   False  False
9998      False       False    False        False      False   False  False
9999      False       False    False        False      False   False  False

      Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0      False    False          False      False           False
1      False    False          False      False           False
2      False    False          False      False           False
3      False    False          False      False           False
4      False    False          False      False           False
...      ...      ...            ...        ...             ...
9995   False    False          False      False           False
9996   False    False          False      False           False
9997   False    False          False      False           False
9998   False    False          False      False           False
9999   False    False          False      False           False

      EstimatedSalary  Exited
0               False   False
1               False   False
2               False   False
3               False   False
4               False   False
...               ...     ...
9995            False   False
9996            False   False
9997            False   False
9998            False   False
9999            False   False

[10000 rows x 14 columns]
```

```
> df.isnull().sum()
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
> df.isnull().sum()/df.shape[0]*100
RowNumber          0.0
CustomerId         0.0
Surname            0.0
CreditScore        0.0
Geography          0.0
Gender             0.0
Age                0.0
Tenure             0.0
Balance            0.0
NumOfProducts      0.0
HasCrCard          0.0
IsActiveMember     0.0
EstimatedSalary    0.0
Exited             0.
```

```
y_pred = lr_ds.predict(X_prob[X_test])
y_pred_prob = lr_ds.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
print('AUC: %f' % roc_auc)
plt.figure(figsize=(12,5))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label=f'Model (AUC = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
AUC: 0.748
```

```
from sklearn.metrics import classification_report
```

```
y = df['Outcome']
X = df.drop('Outcome', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       107
           1       0.71      0.60      0.65        47

    accuracy                           0.77       154
   macro avg       0.75      0.73      0.74       154
weighted avg       0.76      0.77      0.76       154
```

```
y = df['Outcome']
X = df.drop('Outcome', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_pred_prob = lr.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
print('AUC: %f' % roc_auc)
plt.figure(figsize=(12,5))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label=f'Model (AUC = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
AUC: 0.748
```

```
y = df['Outcome']
X = df.drop('Outcome', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_pred_prob = lr.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
print('AUC: %f' % roc_auc)
plt.figure(figsize=(12,5))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label=f'Model (AUC = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
AUC: 0.748
```




```
y = df['Outcome']
X = df.drop('Outcome', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_pred_prob = lr.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
print('AUC: %f' % roc_auc)
plt.figure(figsize=(12,5))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label=f'Model (AUC = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
AUC: 0.748
```




```
y = df['Outcome']
X = df.drop('Outcome', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_pred_prob = lr.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
print('AUC: %f' % roc_auc)
plt.figure(figsize=(12,5))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label=f'Model (AUC = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show
```

```
AUC: 0.748
```




```
y = df['Outcome']
X = df.drop('Outcome', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_pred_prob = lr.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
print('AUC: %f' % roc_auc)
plt.figure(figsize=(12,5))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label=f'Model (AUC = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show
```

```
AUC: 0.748
```




### Summary of training at least three different classifier models

As I decide to applying Cross-Val technique to classify dataset and then tried some feat and efficient tuning methods.
Here I'll take as with better model.

### A paragraph explaining which of your classifier models you recommend as a final model

Although accuracy is from not one of the algorithm performed very well in terms of F1 score then we can use Logistic Regression.
Because of the data is imbalanced, I decide to select recommended Logistic Regression because it also better for recall and precision.
Because in medical dataset, we don't want to miss any data.

### Summary Key Findings and Insights, which walks your reader through the main drivers of your models.

Glucose is most feat and strong feat for good strength. It varies because of highly imbalanced data. So
Glucose is more chances that model will predict strongly about not able to pay back that loan because
Glucose.
Feat number and value of Fetus_pass

### Suggestions for real steps in analyzing this data, which may include suggesting revisiting this model after adding specific data features that may help you achieve stakeh explanation or a better prediction.

Because of imbalanced data, I'm suggesting to use Cross-Validation technique and Stratified Sampling technique and use feat and boost and try combination of over and under sampling technique like
SMOTE
Tomelinks