```
# Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# Loading the data
df = pd.read_csv('/kaggle/input/financial-financial-distribution-dataset/')
```

```
df.head()
```

	id	gender	age	salary	educlevel	jobcat	minority	jobtime	prevexp	jobtenure
0	628	male	41	57000	15	security officer	no	90	144	36
1	630	female	27	28875	15	clerical	no	27	36	36
2	632	female	37	26250	12	clerical	no	79	381	36
3	633	male	24	30000	12	clerical	yes	135	190	36
4	635	male	28	28500	12	clerical	no	67	214	36

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 id          1000 non-null   int64
 gender      1000 non-null   object
 age         1000 non-null   int64
 salary      1000 non-null   int64
 educlevel   1000 non-null   int64
 jobcat      1000 non-null   object
 minority    1000 non-null   object
 jobtime     1000 non-null   int64
 prevexp     1000 non-null   int64
 jobtenure   1000 non-null   int64
dtypes: int64(7), object(3)
memory usage: 78.2+ KB
```

```
df.describe()
```

	id	age	salary	educlevel	jobtime	prevexp	jobtenure
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	1019.000000	31.689000	34418.808000	13.491000	81.903000	95.851000	85.774000
std	290.824949	9.077650	17075.661465	2.884417	8.100468	104.579180	26.777784
min	628.000000	18.000000	15750.000000	8.000000	63.000000	0.000000	36.000000
25%	773.750000	25.000000	24000.000000	12.000000	75.000000	12.750000	63.000000
50%	1019.000000	31.000000	28875.000000	15.000000	81.000000	57.000000	84.000000
75%	1264.250000	37.000000	36000.000000	15.000000	90.000000	144.000000	108.000000
max	1509.000000	51.000000	135000.000000	21.000000	98.000000	476.000000	108.000000

```
df.shape
```

```
(1000, 10)
```

```
df.isnull().sum()
```

```
id            0
gender        0
age           0
salary        0
educlevel     0
jobcat        0
minority      0
jobtime       0
prevexp       0
jobtenure     0
dtype: int64
```

Data cleaning

```
# Dropping unwanted features
df.drop('id',axis=1,inplace=True)
df.drop('jobcat',axis=1,inplace=True)
df.drop('minority',axis=1,inplace=True)
```

```
# Checking the shape after dropping
df.shape
```

```
(1000, 7)
```

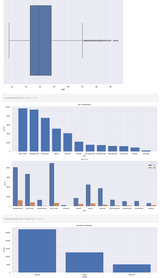
Exploratory Data Analysis

```
def univariateNumerical(df,feature,):
    print('Skewness: {0}'.format(df[feature].skew()))
    print('Kurtosis: {0}'.format(df[feature].kurt()))
    print('Mean: {0}'.format(df[feature].mean()))
    print('Median: {0}'.format(df[feature].median()))
    print('Mode: {0}'.format(df[feature].mode()[0]))
    print('Range: {0}'.format(df[feature].max()-df[feature].min()))
    print('Interquartile: {0}'.format(df[feature].quantile(0.75)-df[feature].quantile(0.25)))

    plt.subplots(2,2,)
    plt.suptitle('univariate Numerical',fontsize=20,)
    plt.subplot(2,2,1,)
    plt.title('Histogram',)
    sns.distplot(df[feature],bins=20,color='r',kde=False,)
    plt.subplot(2,2,2,)
    plt.title('Boxplot',)
    sns.boxplot(df[feature],color='r',orient='v',)
    plt.subplot(2,2,3,)
    plt.title('Density Plot',)
    sns.distplot(df[feature],hist=False,color='r',kde_kws={'shade':True},)
    plt.subplot(2,2,4,)
    plt.title('Cumulative Density Plot',)
    sns.distplot(df[feature],hist_kws={'cumulative':True},kde=False,color='r',)
    plt.show()
```

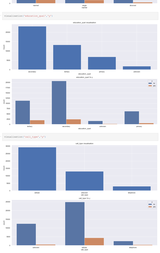
```
univariateNumerical(df,'age')
```

```
Skewness: 0.8827154744231362
Kurtosis: 0.4168922207992167
Mean: 31.689
Median: 31.0
Mode: 28
Range: 33
Interquartile: 12.0
```




```
univariateNumerical(df,'salary',)
```

```
Skewness: 2.117276179991798
Kurtosis: 6.755915722967767
Mean: 34418.808
Median: 28875.0
Mode: 15750
Range: 119250
Interquartile: 12000.0
```

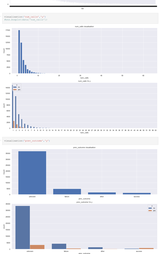



```
univariateNumerical(df,'educlevel',)
```

```
Skewness: 0.43369201133179395
Kurtosis: -0.027837960835052343
Mean: 13.491
Median: 15.0
Mode: 12
Range: 13
Interquartile: 3.0
```

```
univariateNumerical(df,'jobtime',)
```

```
Skewness: -0.04232747112602471
Kurtosis: 0.2995754992818037
Mean: 81.903
Median: 81.0
Mode: 78
Range: 35
Interquartile: 15.0
```




```
univariateNumerical(df,'prevexp',)
```

```
Skewness: 1.1167009471923706
Kurtosis: 0.7269938650306748
Mean: 95.851
Median: 57.0
Mode: 12
Range: 476
Interquartile: 131.25
```

```
univariateNumerical(df,'jobtenure',)
```

```
Skewness: -0.3046401839501297
Kurtosis: -1.2242887231968097
Mean: 85.774
Median: 84.0
Mode: 108
Range: 72
Interquartile: 45.0
```

```
# Gender distribution
df.gender.value_counts
```

```
<bound_method>
gender
female    474
male      526
Name: count, dtype: int64
```

```
# Educational level distribution
df.educlevel.value_counts
```

```
educlevel
15    258
12    249
16    151
14    117
13    102
8      74
17     49
18     00
21     00
Name: count, dtype: int64
```

```
# Job category distribution
df.jobcat.value_counts
```

```
jobcat
clerical            363
security officer    157
manager             157
Name: count, dtype: int64
```

```
# Minority status distribution
df.minority.value_counts
```

```
minority
no     474
yes    526
Name: count, dtype: int64
```

```
# Job time distribution
df.jobtime.value_counts
```

```
jobtime
90    150
81    150
78    150
84    150
75    150
63    150
98    150
Name: count, dtype: int64
```

```
# Previous experience distribution
df.prevexp.value_counts
```

```
prevexp
12    15
0     14
57    13
36    12
144   12
      ..
476    1
Name: count, dtype: int64
```

```
# Job tenure distribution
df.jobtenure.value_counts
```

```
jobtenure
108    150
84     150
72     150
63     150
36     150
Name: count, dtype: int64
```

```
# Salary distribution
df.salary.value_counts
```

```
salary
28875    20
30000    19
27000    18
24000    17
26250    16
         ..
31500     1
31875     1
32250     1
32625     1
33000     1
Name: count, dtype: int64
```

```
# Age distribution
df.age.value_counts
```

```
age
28    100
25     90
27     84
26     81
29     78
30     75
31     63
32     54
33     48
34     42
35     36
36     30
37     24
38     18
39     12
40     09
41     06
42     03
43     00
44     00
45     00
46     00
47     00
48     00
49     00
50     00
51     00
18     00
Name: count, dtype: int64
```

```
# Gender distribution
df.gender.value_counts
```

```
```

[52] ⏱ Visualizing Feature Importance

- **Important Features for Customer Conversion in Duration: Age,day and month**

[53] ⏱ # Calculate and sort feature importances
```
importances = dt_model.feature_importances_
```

[54] ⏱ dt_feature_importance_df = pd.DataFrame({
```
    'Feature': X_train.columns,
    'Importance': importances
})
```

[55] ⏱ dt_feature_importance_df = dt_feature_importance_df.sort_values('Importance', ascending=False)
```
dt_feature_importance_df.head(10)
```

[56] ⏱ plt.figure(figure=(20, 10))
```
plot_tree(dt_model,
          feature_names=X_train.columns,
          class_names=['No Subscription', 'Subscription'],
          filled=True,
          rounded=True,
          max_depth=3,
          fontsize=10)
plt.title('Decision Tree Visualization (Top 3 Levels)', fontsize=20)
plt.show()
```

Random Forest Classifier Model

[57] ⏱ # Random Forest Classifier
```
rf = RandomForestClassifier(n_estimators=100, max_depth=5 ,n
```

[58] ⏱ rf.fit(X_train, y_train)

[59] ⏱ y_pred_rf = rf.predict(X_test)
```
y_pred_rf
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

[60] ⏱ # Calculate and print accuracy
```
accuracy = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", accuracy)
print("\nRandom Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.9
Random Forest Classification Report:
              precision    recall  f1-score   support
```

Conclusion

ML Model AUROC Score

- Gaussian: **87%**
- Random Forest: **90%**
- Logistic Regression: **89%**
- Decision Tree: **87%**
- KNN Model: **88%**

- **Important Features for Customer Conversion in Duration: Age,day and month**

[61] ⏱ # Get feature importance for Random Forest
```
rf_importance = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': rf.feature_importances_
})
```

Visualization

[62] ⏱