PREDICTING THE SUCCESS OF BANK MARKETING CAMPAIGN

IMPORTING NECESSARY PACKAGES AND MODULES

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler,LabelEncoder
from sklearn.model_selection import train_test_split,GridSearchCV
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report,ConfusionMatrixDisplay,accuracy_score,auc,roc_curve,RocCurveDisplay
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pickle
import warnings
warnings.filterwarnings('ignore')
```

Importing the Dataset

```
df=pd.read_csv('/content/drive/MyDrive/Bank_Marketing_Prediction.csv')
df
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previo
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	1	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	1	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	1	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	1	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	1	
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	1	
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	1	
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	

45211 rows × 17 columns

Checking The Shape of Dataset

```
df.shape
```

```
(45211, 17)
```

Describing the Dataset

```
df.describe()
```

	Age	Account balance	Day of campaign	Call duration	No. of Campaign	No of days passed after campaign	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1415.196081	15.806419	258.163080	2.763841	41.832563	0.580323
std	10.618762	3020.529906	8.322476	257.527812	3.098021	99.456849	2.303441
min	18.000000	0.000000	1.000000	0.000000	1.000000	1.000000	0.000000
25%	33.000000	137.000000	8.000000	103.000000	1.000000	1.000000	0.000000
50%	39.000000	485.000000	16.000000	180.000000	2.000000	1.000000	0.000000
75%	48.000000	1436.000000	21.000000	319.000000	3.000000	1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Number of Elements

```
df.size
```

```
768587
```

Checking for Missing Values

```
df.isna().sum()
```

	0
Age	0
Job	0
Marital status	0
Education	0
Loan default	0
Account balance	0
Housing	0
Loan	0
Contact details	0
Day of campaign	0
Month of campaign	0
Call duration	0
No. of Campaign	0
No of days passed after campaign	0
previous	0
Outcome of previous campaign	0
Outcome of present campaign	0

```
df['Outcome of present campaign'].value_counts()
```

	count
Outcome of present campaign	
no	39922
yes	5289

Checking the unique values in each column

```
cols=['Age','Job','Marital status','Education','Loan default','Account balance','Housing','Loan','Contact details','Day of campaign','Month of cam

for col in cols:
  print('-'*25,col,'-'*25)
  print(df[col].value_counts())
  print('*'*50)
```

```
------------------------- Age -------------------------
Age
32    2085
31    1996
33    1972
34    1930
35    1894
      ...
93       2
90       2
95       2
88       2
94       1
Name: count, Length: 77, dtype: int64
**************************************************
------------------------- Job -------------------------
Job
blue-collar      9732
management       9458
technician       7597
admin.           5171
```

```
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: count, dtype: int64
**************************************************
------------------------- Marital status -------------------------
Marital status
married     27214
single      12790
divorced     5207
Name: count, dtype: int64
**************************************************
------------------------- Education -------------------------
Education
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: count, dtype: int64
**************************************************
------------------------- Loan default -------------------------
Loan default
no     44396
yes      815
Name: count, dtype: int64
**************************************************
------------------------- Account balance -------------------------
Account balance
0        3514
1         245
2         181
3         156
```

Dealing Missing Values

Here we can see in many columns there are some values as unknown. So we should consider this as missing values and treat them so.

```
# DROPING THE COLUMN " OUTC0ME OF PREVIOUS CAMPAIGN"  BCZ THERE IS LARGE NO OF UNKNOWN VALUES

df.drop(['Outcome of previous campaign'],axis=1,inplace=True)
```

Change the unknown as NAN using numpy

```
features=['Job','Education','Contact details']

for feature in features:
  df[feature]=df[feature].replace('unknown',np.nan)

df
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previo
0	58	management	married	tertiary	no	2143	yes	no	NaN	5	may	261	1	1	
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151	1	1	
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76	1	1	
3	47	blue-collar	married	NaN	no	1506	yes	no	NaN	5	may	92	1	1	
4	33	NaN	single	NaN	no	1	no	no	NaN	5	may	198	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	1	
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	1	
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	1	
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	

45211 rows × 16 columns

Filling the Missing Values

```
features=['Job','Education','Contact details']

for feature in features:
  df[feature]=df[feature].fillna(df[feature].mode()[0])

df
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previo
0	58	management	married	tertiary	no	2143	yes	no	cellular	5	may	261	1	1	
1	44	technician	single	secondary	no	29	yes	no	cellular	5	may	151	1	1	
2	33	entrepreneur	married	secondary	no	2	yes	yes	cellular	5	may	76	1	1	
3	47	blue-collar	married	secondary	no	1506	yes	no	cellular	5	may	92	1	1	
4	33	blue-collar	single	secondary	no	1	no	no	cellular	5	may	198	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	1	
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	1	
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	1	
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	

45211 rows × 16 columns

Again checking the if there is any missing left

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

	0
Age	0
Job	0
Marital status	0
Education	0
Loan default	0
Account balance	0
Housing	0
Loan	0
Contact details	0
Day of campaign	0
Month of campaign	0
Call duration	0
No. of Campaign	0
No of days passed after campaign	0
previous	0
Outcome of present campaign	0

Checking Datatype of Attributes

```
df.dtypes
```

	0
Age	int64
Job	object
Marital status	object
Education	object
Loan default	object
Account balance	int64
Housing	object
Loan	object
Contact details	object
Day of campaign	int64
Month of campaign	object
Call duration	int64
No. of Campaign	int64
No of days passed after campaign	int64
previous	int64
Outcome of present campaign	object

```
df1=df.copy()
```

Converting the Datatype of features from string to numeric for preprocessing

Label Encoding the Data

```
encoder=LabelEncoder()

cols=['Job','Marital status','Education','Loan default','Housing','Loan','Contact details','Month of campaign','Outcome of present campaign']

for col in cols:
  df[col]=encoder.fit_transform(df[col])

df
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previous	Outc pres campa
0	58	4	1	2	0	2143	1	0	0	5	8	261	1	1	0	
1	44	9	2	1	0	29	1	0	0	5	8	151	1	1	0	
2	33	2	1	1	0	2	1	1	0	5	8	76	1	1	0	
3	47	1	1	1	0	1506	1	0	0	5	8	92	1	1	0	
4	33	1	2	1	0	1	0	0	0	5	8	198	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	1	0	
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	1	0	
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3	
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	1	0	
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11	

45211 rows × 16 columns

```
df.to_csv('df.csv',index=False)
from google.colab import files
files.download('df.csv')
```

```
df.dtypes
```

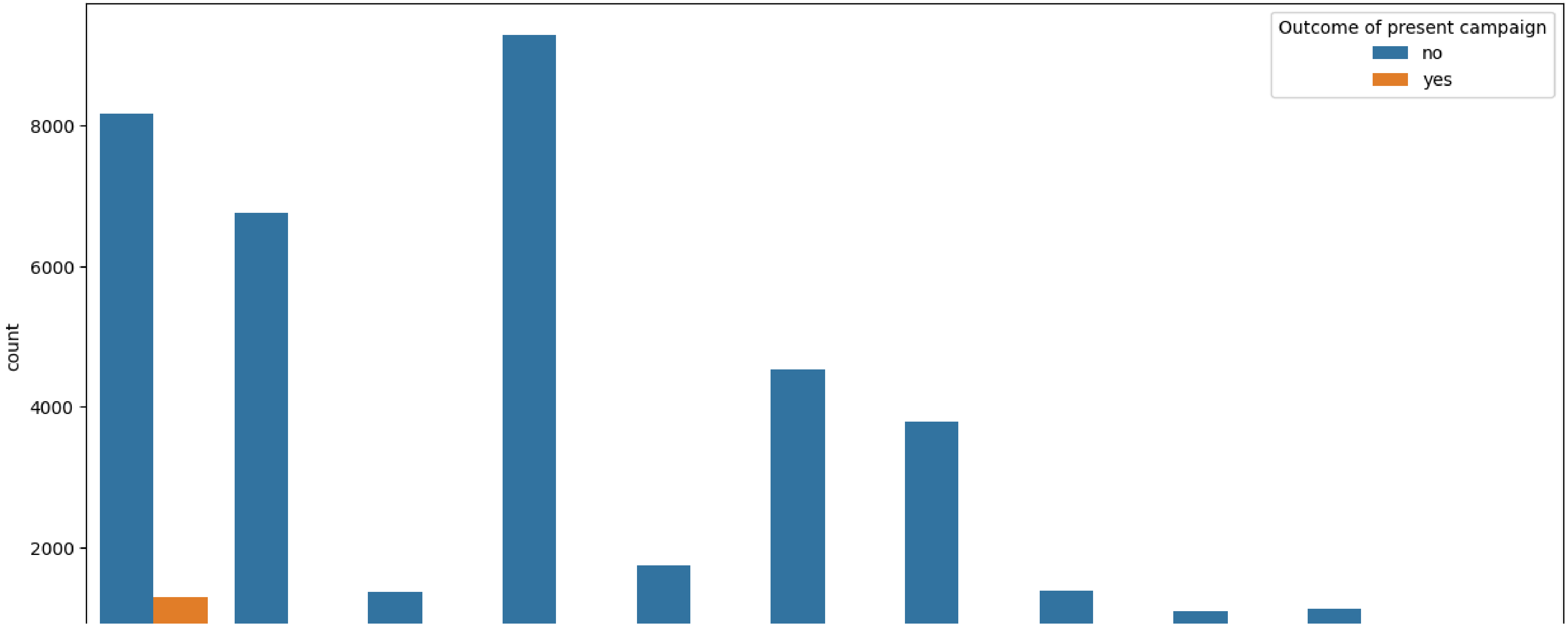
	0
Age	int64
Job	int64
Marital status	int64
Education	int64
Loan default	int64
Account balance	int64
Housing	int64
Loan	int64
Contact details	int64
Day of campaign	int64
Month of campaign	int64
Call duration	int64
No. of Campaign	int64
No of days passed after campaign	int64
previous	int64
Outcome of present campaign	int64

Visualizations of the dataset

Count Plot of Job with class label as hue

This plot helps us to identify people belonging to which job group opted more for term deposit. We can see that people belonging to Management job opted more for term deposit

```
plt.figure(figsize=(15,7))
sns.countplot(x='Job',data=df1,hue='Outcome of present campaign')
plt.show()
```



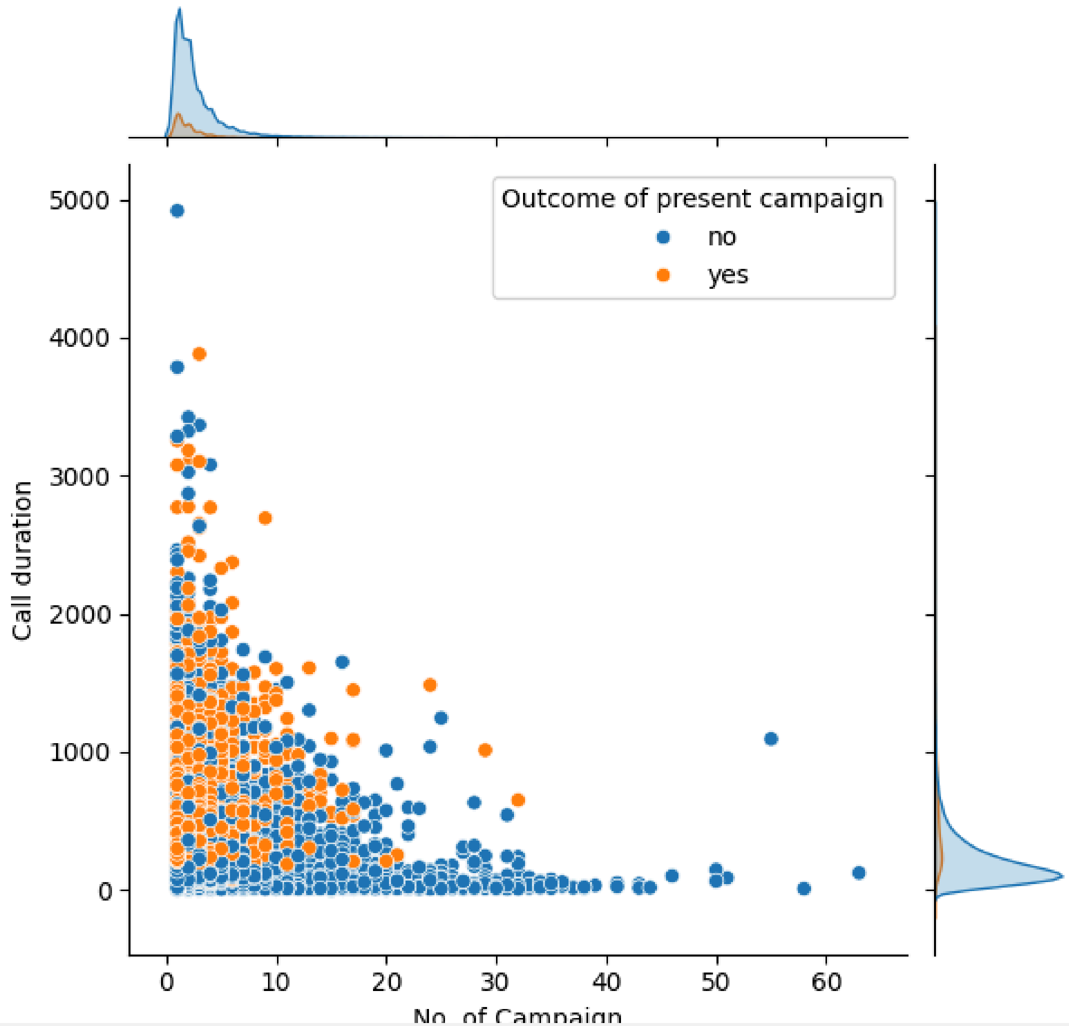
Joint plot of campaign V/s Duration of call with class label as hue

Job

The plot shows that the call duration decreased with the increase in no of campaigns and there is no significant change in the outcome of present campaign with the increase in the no of campaign

```
sns.jointplot(x='No. of Campaign', y='Call duration',data=df1,hue='Outcome of present campaign')
```

<seaborn.axisgrid.JointGrid at 0x7b4db187a680>

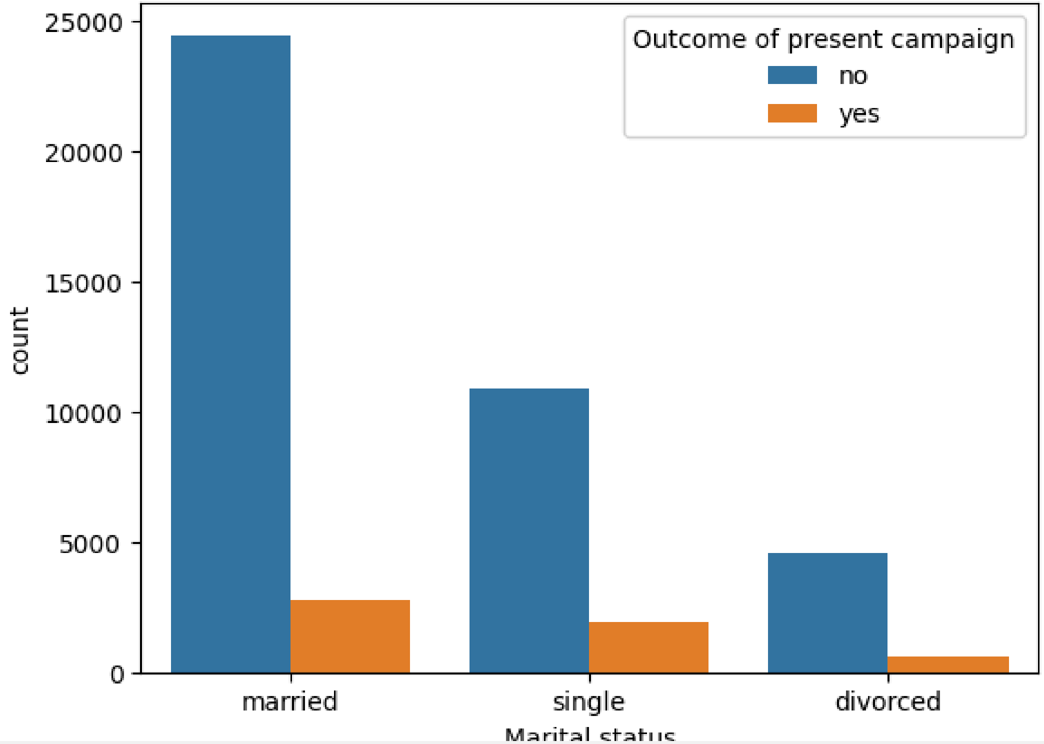


Count plot of Marital status with Class label as hue

This plot helps us to identify people belonging to which marital group opted more for term deposit We can see that married people opted more for term deposit

```
sns.countplot(x='Marital status',data=df1,hue='Outcome of present campaign')
```

```
<Axes: xlabel='Marital status', ylabel='count'>
```

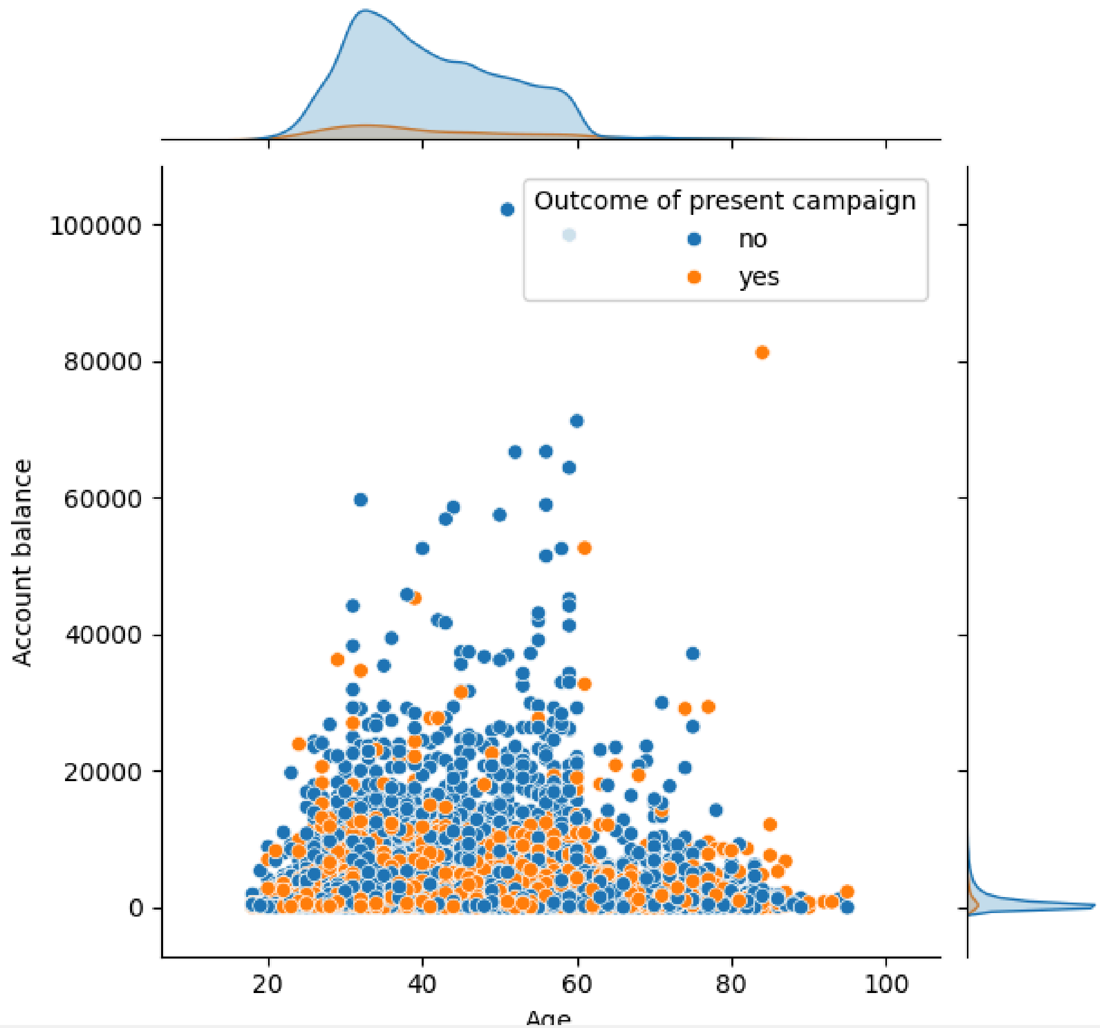


Joint plot of Age vs Account Balance with class label as hue

This plot shows that people from different age group even if they have high balance in their account they chose not to opt in for term deposit

```
sns.jointplot(x='Age',y='Account balance',data=df1,hue='Outcome of present campaign')
```

```
<seaborn.axisgrid.JointGrid at 0x7b4d95efd870>
```



Splitting Features and Class Label

```
X=df.iloc[:,:-1]
X
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previous
0	58	4	1	2	0	2143	1	0	0	5	8	261	1	1	0
1	44	9	2	1	0	29	1	0	0	5	8	151	1	1	0
2	33	2	1	1	0	2	1	1	0	5	8	76	1	1	0
3	47	1	1	1	0	1506	1	0	0	5	8	92	1	1	0
4	33	1	2	1	0	1	0	0	0	5	8	198	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	1	0
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	1	0
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	1	0
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11

```
y=df.iloc[:,-1]
y
```

	Outcome of present campaign
0	0
1	0
2	0
3	0
4	0
...	...
45206	1
45207	1
45208	1
45209	0
45210	0

45211 rows × 1 columns

```
# sns.jointplot(x='age',y='balance',data=df,hue='y')
# plt.show()
```

Scaling the X values

```
scaler=MinMaxScaler()
X_scaled=scaler.fit_transform(X)
X_scaled

# bcoz when we didnot scale the values there will be a chance for prediction can be more depend on some features
```

```
array([[0.51948052, 0.4       , 0.5       , ..., 0.        , 0.        ,
        0.        ],
       [0.33766234, 0.9       , 1.        , ..., 0.        , 0.        ,
        0.        ],
       [0.19480519, 0.2       , 0.5       , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.7012987 , 0.5       , 0.5       , ..., 0.06451613, 0.21034483,
        0.01090909],
       [0.50649351, 0.1       , 0.5       , ..., 0.0483871 , 0.        ,
        0.        ],
       [0.24675325, 0.2       , 0.5       , ..., 0.01612903, 0.21494253,
        0.04      ]])
```

```
df
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previous	Outc pres campa
0	58	4	1	2	0	2143	1	0	0	5	8	261	1	1	0	
1	44	9	2	1	0	29	1	0	0	5	8	151	1	1	0	
2	33	2	1	1	0	2	1	1	0	5	8	76	1	1	0	
3	47	1	1	1	0	1506	1	0	0	5	8	92	1	1	0	
4	33	1	2	1	0	1	0	0	0	5	8	198	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	1	0	
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	1	0	
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3	
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	1	0	
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11	

45211 rows × 16 columns

Splitting the Values for Training and Testing

```
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,random_state=1,test_size=.3)
```

Creating different ML Models and Evaluating its performance

Fitting the dataset , making prediction and checking performance using classification_report and Confusion Matrix Display

```
knn=KNeighborsClassifier(n_neighbors=3)
svc=SVC()
dt=DecisionTreeClassifier()
rf=RandomForestClassifier(random_state=1)
nb=GaussianNB()
ab=AdaBoostClassifier(random_state=1)
gb=GradientBoostingClassifier(random_state=1)
xg=XGBClassifier(random_state=1)

models=[knn,svc,dt,rf,nb,ab,gb,xg]

for model in models:
       model.fit(X_train,y_train)
       y_pred=model.predict(X_test)
       print(model)
       print(classification_report(y_test,y_pred))
       print('*'*60)
       #print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
KNeighborsClassifier(n_neighbors=3)
              precision    recall  f1-score   support
```

```
           0       0.91      0.97      0.94     12013
           1       0.48      0.22      0.30      1551

    accuracy                           0.88     13564
   macro avg       0.69      0.59      0.62     13564
weighted avg       0.86      0.88      0.86     13564

************************************************************
SVC()
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     12013
           1       0.58      0.02      0.04      1551

    accuracy                           0.89     13564
   macro avg       0.73      0.51      0.49     13564
weighted avg       0.85      0.89      0.84     13564

************************************************************
DecisionTreeClassifier()
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     12013
           1       0.41      0.45      0.43      1551

    accuracy                           0.86     13564
   macro avg       0.67      0.68      0.68     13564
weighted avg       0.87      0.86      0.87     13564

************************************************************
RandomForestClassifier(random_state=1)
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     12013
           1       0.59      0.36      0.45      1551

    accuracy                           0.90     13564
   macro avg       0.76      0.67      0.70     13564
weighted avg       0.88      0.90      0.89     13564

************************************************************
GaussianNB()
              precision    recall  f1-score   support
```

```
           0       0.92      0.92      0.92     12013
           1       0.39      0.42      0.40      1551

    accuracy                           0.86     13564
   macro avg       0.66      0.67      0.66     13564
weighted avg       0.86      0.86      0.86     13564

************************************************************
AdaBoostClassifier(random_state=1)
              precision    recall  f1-score   support
```

Checking the correlation of Features with Class label

```
df.corr()
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Contact details	Day of campaign	Month of campaign	Call duration	No. of Campaign
Age	1.000000	-0.034420	-0.403240	-0.164888	-0.017879	0.097475	-0.185513	-0.015655	0.170349	-0.009120	-0.042357	-0.004648	0.004760
Job	-0.034420	1.000000	0.062377	0.184084	-0.005285	0.015243	-0.108219	-0.025496	-0.010252	0.025840	-0.090921	0.006361	0.003448
Marital status	-0.403240	0.062377	1.000000	0.119220	-0.007023	-0.000487	-0.016096	-0.046893	-0.020524	-0.005261	-0.006991	0.011852	-0.008994
Education	-0.164888	0.184084	0.119220	1.000000	-0.011539	0.066838	-0.075157	-0.025282	-0.070190	0.025931	-0.075052	0.002635	0.003703
Loan default	-0.017879	-0.005285	-0.007023	-0.011539	1.000000	-0.047285	-0.006025	0.077234	-0.017208	0.009424	0.011486	-0.010021	0.016822
Account balance	0.097475	0.015243	-0.000487	0.066838	-0.047285	1.000000	-0.061628	-0.074512	0.035904	0.007073	0.023312	0.021043	-0.013718
Housing	-0.185513	-0.108219	-0.016096	-0.075157	-0.006025	-0.061628	1.000000	0.041323	-0.080822	-0.027982	0.271481	0.005075	-0.023599
Loan	-0.015655	-0.025496	-0.046893	-0.025282	0.077234	-0.074512	0.041323	1.000000	-0.013183	0.011370	0.022145	-0.012412	0.009980
Contact details	0.170349	-0.010252	-0.020524	-0.070190	-0.017208	0.035904	-0.080822	-0.013183	1.000000	0.023652	-0.004616	-0.023201	0.053895
Day of campaign	-0.009120	0.025840	-0.005261	0.025931	0.009424	0.007073	-0.027982	0.011370	0.023652	1.000000	-0.006028	-0.030206	0.162490
Month of campaign	-0.042357	-0.090921	-0.006991	-0.075052	0.011486	0.023312	0.271481	0.022145	-0.004616	-0.006028	1.000000	0.006314	-0.110031
Call duration	-0.004648	0.006361	0.011852	0.002635	-0.010021	0.021043	0.005075	-0.012412	-0.023201	-0.030206	0.006314	1.000000	-0.084570
No. of Campaign	0.004760	0.003448	-0.008994	0.003703	0.016822	-0.013718	-0.023599	0.009980	0.053895	0.162490	-0.110031	-0.084570	1.000000
No of days passed after campaign	-0.023924	-0.021088	0.019108	0.003977	-0.029875	0.001812	0.124523	-0.022665	0.015933	-0.093003	0.033042	-0.001603	-0.088387
previous	0.001288	0.001307	0.014973	0.025175	-0.018329	0.015324	0.037076	-0.011043	0.028097	-0.051710	0.022727	0.001203	-0.032855

Outcome of present campaign	0.025155	0.040786	0.045588	0.068633	-0.022419	0.049783	-0.139173	-0.068185	0.014042	-0.028348	-0.024471	0.394521	-0.073172

Heat Map for Correlation

```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=True)
```

<Axes: >


Age
Job
Marital status
Education
Loan default
Account balance
Housing
Loan
Contact details
Day of campaign
Month of campaign
Call duration
No. of Campaign
1 -0.034 -0.4 -0.16 -0.018 0.097 -0.19 -0.016 0.17 -0.0091 -0.042 -0.0046 0.0048 -0.024 0.0013 0.025
-0.034 1 0.062 0.18 -0.0053 0.015 -0.11 -0.025 -0.01 0.026 -0.091 0.0064 0.0034 -0.021 0.0013 0.041
-0.4 0.062 1 0.12 -0.007 -0.00049 -0.016 -0.047 -0.021 -0.0053 -0.007 0.012 -0.009 0.019 0.015 0.046
-0.16 0.18 0.12 1 -0.012 0.067 -0.075 -0.025 -0.07 0.026 -0.075 0.0026 0.0037 0.004 0.025 0.069
-0.018 -0.0053 -0.007 -0.012 1 -0.047 -0.006 0.077 -0.017 0.0094 0.011 -0.01 0.017 -0.03 -0.018 -0.022
0.097 0.015 -0.00049 0.067 -0.047 1 -0.062 -0.075 0.036 0.0071 0.023 0.021 -0.014 0.0018 0.015 0.05
-0.19 -0.11 -0.016 -0.075 -0.006 -0.062 1 0.041 -0.081 -0.028 0.27 0.0051 -0.024 0.12 0.037 -0.14
-0.016 -0.025 -0.047 -0.025 0.077 -0.075 0.041 1 -0.013 0.011 0.022 -0.012 0.01 -0.023 -0.011 -0.068
0.17 -0.01 -0.021 -0.07 -0.017 0.036 -0.081 -0.013 1 0.024 -0.0046 -0.023 0.054 0.016 0.028 0.014
-0.0091 0.026 -0.0053 0.026 0.0094 0.0071 -0.028 0.011 0.024 1 -0.006 -0.03 0.16 -0.093 -0.052 -0.028
-0.042 -0.091 -0.007 -0.075 0.011 0.023 0.27 0.022 -0.0046 -0.006 1 0.0063 -0.11 0.033 0.023 -0.024
-0.0046 0.0064 0.012 0.0026 -0.01 0.021 0.0051 -0.012 -0.023 -0.03 0.0063 1 -0.085 -0.0016 0.0012 0.39
0.0048 0.0034 -0.009 0.0037 0.017 -0.014 -0.024 0.01 0.054 0.16 -0.11 -0.085 1 -0.088 -0.033 -0.073
1.0
0.8
0.6
0.4
0.2
0.0

Drop Features with low correltion with the output

We are dropping a feature as there is no change in accuracy if we drop this features.

```
df.drop(['Contact details'],axis=1,inplace=True)
df
```

	Age	Job	Marital status	Education	Loan default	Account balance	Housing	Loan	Day of campaign	Month of campaign	Call duration	No. of Campaign	No of days passed after campaign	previous	Outcome of present campaign
0	58	4	1	2	0	2143	1	0	5	8	261	1	1	0	0
1	44	9	2	1	0	29	1	0	5	8	151	1	1	0	0
2	33	2	1	1	0	2	1	1	5	8	76	1	1	0	0
3	47	1	1	1	0	1506	1	0	5	8	92	1	1	0	0
4	33	1	2	1	0	1	0	0	5	8	198	1	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	9	1	2	0	825	0	0	17	9	977	3	1	0	1
45207	71	5	0	0	0	1729	0	0	17	9	456	2	1	0	1
45208	72	5	1	1	0	5715	0	0	17	9	1127	5	184	3	1
45209	57	1	1	1	0	668	0	0	17	9	508	4	1	0	0
45210	37	2	1	1	0	2971	0	0	17	9	361	2	188	11	0

Splitting the Features and output after feature selection

```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

Scaling and Splitting the Features for Train and Test

```
X_scaled=scaler.fit_transform(X)

X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,random_state=1,test_size=.3)
```

Checking the shape of both train and test data

```
X_train.shape,y_train.shape
```

```
((31647, 14), (31647,))
```

```
X_test.shape,y_test.shape
```

```
((13564, 14), (13564,))
```

Creating models and Checking Performance

```
knn=KNeighborsClassifier(n_neighbors=3)
svc=SVC()
dt=DecisionTreeClassifier()
rf=RandomForestClassifier(random_state=1)
nb=GaussianNB()
ab=AdaBoostClassifier(random_state=1)
gb=GradientBoostingClassifier(random_state=1)
xg=XGBClassifier(random_state=1)

models=[knn,svc,dt,rf,nb,ab,gb,xg]

for model in models:
       model.fit(X_train,y_train)
       y_pred=model.predict(X_test)
       print(model)
       print(classification_report(y_test,y_pred))
       print('*'*60)
```

```
KNeighborsClassifier(n_neighbors=3)
              precision    recall  f1-score   support

           0       0.91      0.97      0.94     12013
           1       0.48      0.22      0.30      1551

    accuracy                           0.88     13564
   macro avg       0.69      0.60      0.62     13564
weighted avg       0.86      0.88      0.86     13564

************************************************************
SVC()
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     12013
           1       0.59      0.02      0.04      1551

    accuracy                           0.89     13564
   macro avg       0.74      0.51      0.49     13564
weighted avg       0.85      0.89      0.84     13564

************************************************************
DecisionTreeClassifier()
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     12013
```

```
           1       0.41      0.46      0.43      1551

    accuracy                           0.86     13564
   macro avg       0.67      0.69      0.68     13564
weighted avg       0.87      0.86      0.87     13564

************************************************************
RandomForestClassifier(random_state=1)
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     12013
           1       0.60      0.37      0.46      1551

    accuracy                           0.90     13564
   macro avg       0.76      0.67      0.70     13564
weighted avg       0.89      0.90      0.89     13564

************************************************************
GaussianNB()
              precision    recall  f1-score   support

           0       0.92      0.92      0.92     12013
           1       0.40      0.42      0.41      1551

    accuracy                           0.86     13564
   macro avg       0.66      0.67      0.66     13564
weighted avg       0.86      0.86      0.86     13564

************************************************************
AdaBoostClassifier(random_state=1)
              precision    recall  f1-score   support
```

Checking the values in prediction

```
y.value_counts()
```

	count
Outcome of present campaign	
0	39922
1	5289

OVER SAMPLING

Its a imbalanced dataset. So we should make it balanced with over sampling or under sampling. Here we are using oversampling, Bcoz its more appropritate to train with more datapoints for better performance

```
os=SMOTE(random_state=1)
X_os,y_os=os.fit_resample(X,y)
```

Checking the Shape of Over Sampled Dataset

```
X_os.shape,y_os.shape
```

```
((79844, 14), (79844,))
```

Scaling the Oversampled Dataset

```
X_scaled_os=scaler.fit_transform(X_os)
```

Splitting the Oversampled dataset into Train and Test Data

```
X_train_os,X_test_os,y_train_os,y_test_os=train_test_split(X_scaled_os,y_os,random_state=1,test_size=.3)
```

Building model and evaluating Performance for Oversampled Data

```
knn=KNeighborsClassifier(n_neighbors=3)
svc=SVC()
dt=DecisionTreeClassifier()
rf=RandomForestClassifier(random_state=1)
nb=GaussianNB()
ab=AdaBoostClassifier(random_state=1)
gb=GradientBoostingClassifier(random_state=1)
xg=XGBClassifier(random_state=1)
```

```
models=[knn,svc,dt,rf,nb,ab,gb,xg]
acc=[]
for model in models:
       model.fit(X_train_os,y_train_os)
       y_pred_os=model.predict(X_test_os)
       acu=accuracy_score(y_test_os,y_pred_os)
       acc.append(acu*100)
       print(model)
       print(classification_report(y_test_os,y_pred_os))
       print('-'*60)
       # print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
KNeighborsClassifier(n_neighbors=3)
              precision    recall  f1-score   support

           0       0.86      0.88      0.87     12025
           1       0.88      0.85      0.86     11929

    accuracy                           0.87     23954
```

```
   macro avg       0.87      0.87      0.87     23954
weighted avg       0.87      0.87      0.87     23954

------------------------------------------------------------
SVC()
              precision    recall  f1-score   support

           0       0.87      0.85      0.86     12025
           1       0.85      0.87      0.86     11929

    accuracy                           0.86     23954
   macro avg       0.86      0.86      0.86     23954
weighted avg       0.86      0.86      0.86     23954

------------------------------------------------------------
DecisionTreeClassifier()
              precision    recall  f1-score   support

           0       0.89      0.87      0.88     12025
           1       0.87      0.89      0.88     11929

    accuracy                           0.88     23954
   macro avg       0.88      0.88      0.88     23954
weighted avg       0.88      0.88      0.88     23954

------------------------------------------------------------
RandomForestClassifier(random_state=1)
              precision    recall  f1-score   support

           0       0.95      0.89      0.92     12025
           1       0.90      0.95      0.92     11929

    accuracy                           0.92     23954
   macro avg       0.92      0.92      0.92     23954
weighted avg       0.92      0.92      0.92     23954

------------------------------------------------------------
GaussianNB()
              precision    recall  f1-score   support

           0       0.91      0.48      0.62     12025
           1       0.64      0.95      0.77     11929

    accuracy                           0.71     23954
   macro avg       0.77      0.71      0.70     23954
```

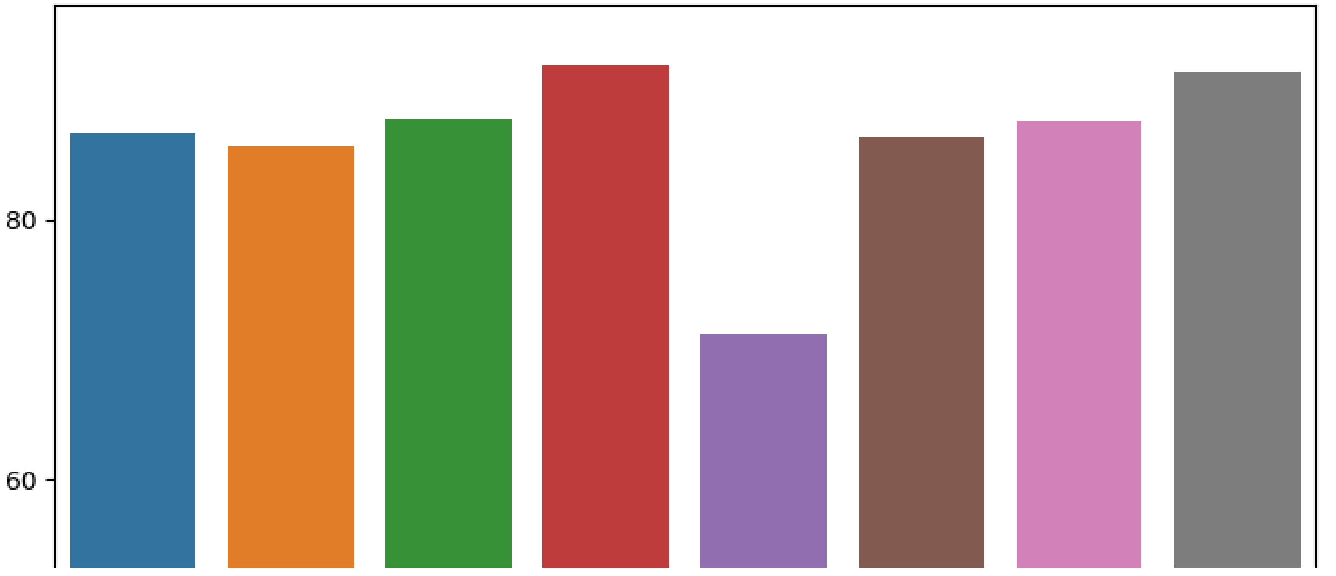
```
weighted avg       0.78      0.71      0.70     23954

------------------------------------------------------------
AdaBoostClassifier(random_state=1)
              precision    recall  f1-score   support
```

```
m=['KNeighborsClassifier()','SVC()','DecisionTreeClassifier()','RandomForestClassifier()','GaussianNB()','AdaBoostClassifier()','GradientBoostingC
acd=pd.DataFrame({'Model':m,'Accuracy':acc})
acd.style.set_properties(**{'background-color': 'red'}, subset=pd.IndexSlice[3, :])
```

	Model	Accuracy
0	KNeighborsClassifier()	86.703682
1	SVC()	85.751858
2	DecisionTreeClassifier()	87.801620
3	RandomForestClassifier()	91.963764
4	GaussianNB()	71.224013
5	AdaBoostClassifier()	86.390582
6	GradientBoostingClassifier()	87.638808
7	XGBClassifier()	91.454454

```
plt.figure(figsize=(9,9))
sns.barplot(x='Model',y='Accuracy',data=acd,hue='Model',legend=False)
plt.xticks(rotation=90)
plt.show()
```


80
60