BANK CUSTOMER CHURN PREDICTION SYSTEM USING ML

INTRODUCTION

Customer Churn prediction means knowing which customers are likely to leave or unsubscribe from your service. For many companies, this is an important prediction. This is because acquiring new customers often costs more than retaining existing ones. Once you've identified customers at risk of churn, you need to know exactly what marketing efforts you should make with each customer to maximize their likelihood of staying.

Customers have different behaviors and preferences, and reasons for cancelling their subscriptions. Therefore, it is important to actively communicate with each of them to keep them on your customer list. You need to know which marketing activities are most effective for individual customers and when they are most effective.

How does Customer Churn Prediction Work?

We first have to do some Exploratory Data Analysis in the Dataset, then fit the dataset into Machine Learning Classification Algorithm and choose the best Algorithm for the Bank Customer Churn Dataset.

ABOUT THE DATASET

This dataset is for ABC Multistate bank with following columns:

✨ customer_id: unused variable. -->Tells about ID of the customer

✨ credit_score: used as input. --> Credit Score of the customer

✨ country: used as input. --> Country of Customer

✨gender: used as input. --> Gender of customer

✨ age: used as input. --> Age of customer

✨ tenure: used as input. --> Tenure of customer

✨ balance: used as input. --> Bank balance of the customer

✨ products_number: used as input. --> how much products the customer have

✨ credit_card: used as input. --> Customer has a credit card or not

✨ active_member: used as input. --> Tells the customer is active member or not

✨ estimated_salary: used as input. --> Estimated Salary of Customer

✨ churn: used as the target. 1 if the client has left the bank during some period or 0 if he/she has not.

In [175]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
sns.set(style="darkgrid")

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score ,f1_score
from sklearn.metrics import confusion_matrix
```

In [176]:

```
df=pd.read_csv("Bank_Churn.csv")
```

BASIC UNDERSTANDING OF DATA

In [177]:

```
df
```

Out[177]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	Has
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	
...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	

10000 rows × 14 columns

In [178]:

```
df.shape
```

Out[178]:

```
(10000, 14)
```

In [179]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [180]:

```
df.describe()
```

Out[180]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000

In [181]:

```
df.describe(include=object)
```

Out[181]:

	Surname	Geography	Gender
count	10000	10000	10000
unique	2932	3	2
top	Smith	France	Male
freq	32	5014	5457

In [182]:

```
df.isnull().sum()/len(df)*100
```

Out[182]:

```
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             0.0
dtype: float64
```

In [183]:

```
df.duplicated().sum()/len(df*100)
```

Out[183]:

```
0.0
```

In [184]:

```
df.head()
```

Out[184]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

DATA PREPROCESSING

In [185]:

```
df.head()
```

Out[185]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

In [186]:

```
# 1. droping irrelevant features of data
```

In [187]:

```
df.drop(columns=['RowNumber','CustomerId','Surname'],inplace=True)
```

In [188]:

```
df
```

Out[188]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
0	619	France	Female	42	2	0.00	1	1	1	10134
1	608	Spain	Female	41	1	83807.86	1	0	1	11254
2	502	France	Female	42	8	159660.80	3	1	0	11393
3	699	France	Female	39	1	0.00	2	0	0	9382
4	850	Spain	Female	43	2	125510.82	1	1	1	7908
...	...	...	...	...	...	...	...	...	...	
9995	771	France	Male	39	5	0.00	2	1	0	9627
9996	516	France	Male	35	10	57369.61	1	1	1	10169
9997	709	France	Female	36	7	0.00	1	0	1	4208
9998	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	792	France	Female	28	4	130142.79	1	1	0	3819

10000 rows × 11 columns

In [189]:

```
#2 changing name of exited feature as churned
```

In [190]:

```
df.rename(columns={"Exited":"Churned"},inplace=True)
```

In [191]:

```
df.head()
```

Out[191]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

EXPLORATORY DATA ANALYSIS

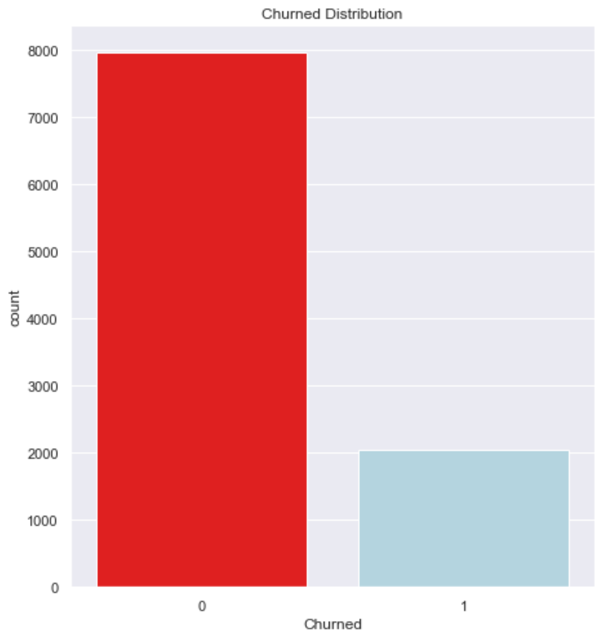
In [192]:

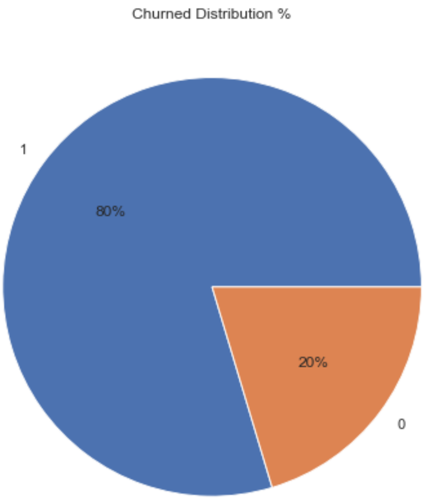
```
# 1) Visualizing Churned Feature:
```

In [193]:

```
plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
plt.title("Churned Distribution")
sns.countplot(df['Churned'],palette=['red','Lightblue'])

plt.subplot(1,2,2)
plt.title("Churned Distribution %")
plt.pie(df['Churned'].value_counts(),labels=df['Churned'].unique(),autopct="%0.0f%%");
```



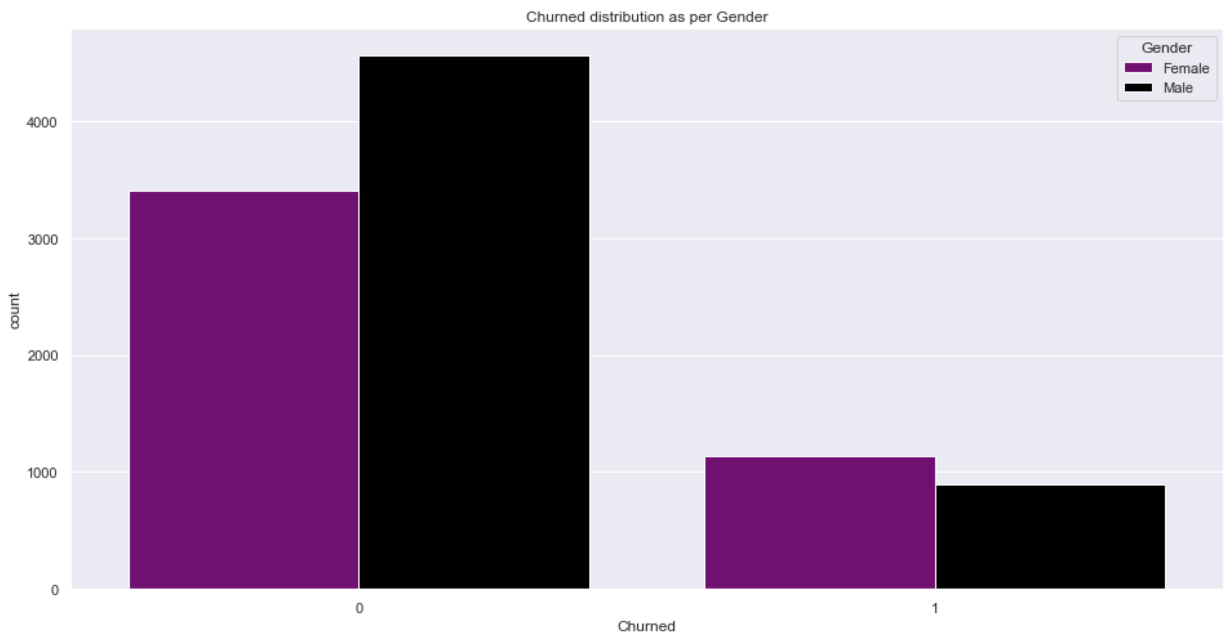


In [194]:

```
# 2) churned as per gender
```

In [195]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per Gender")
sns.countplot(df['Churned'],hue='Gender',palette=['purple','black'],data=df);
```

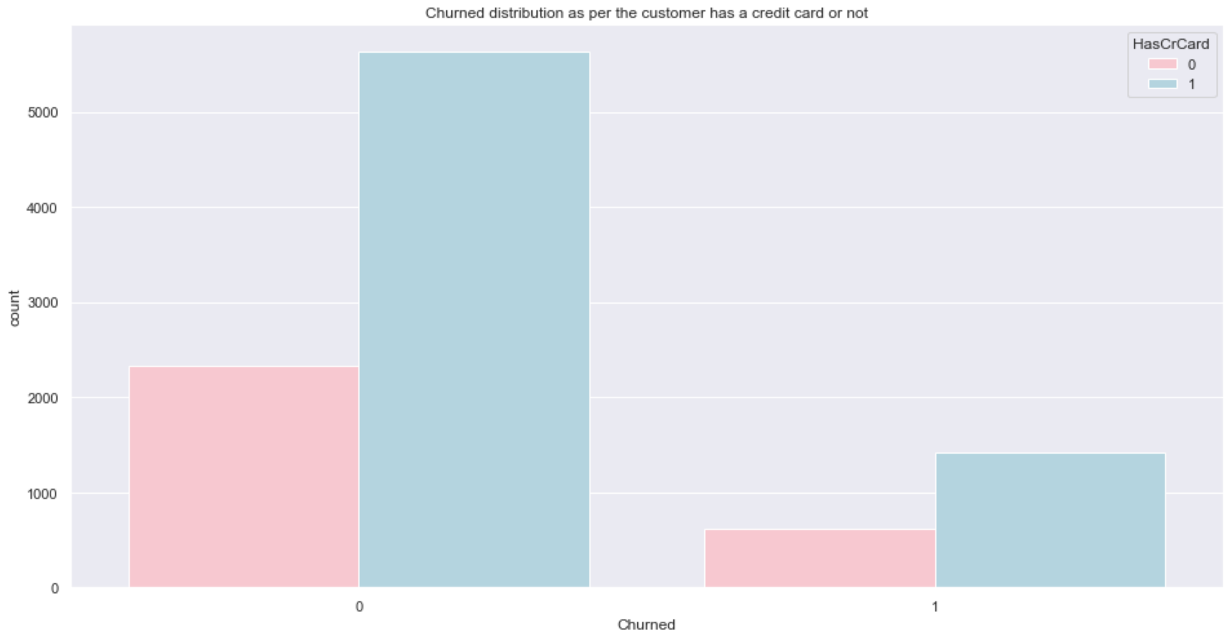


In [196]:

```
# 3) churned as per person has a credit card or not
```

In [197]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per the customer has a credit card or not")
sns.countplot(df['Churned'],hue='HasCrCard',palette=['pink','lightblue'],data=df);
```

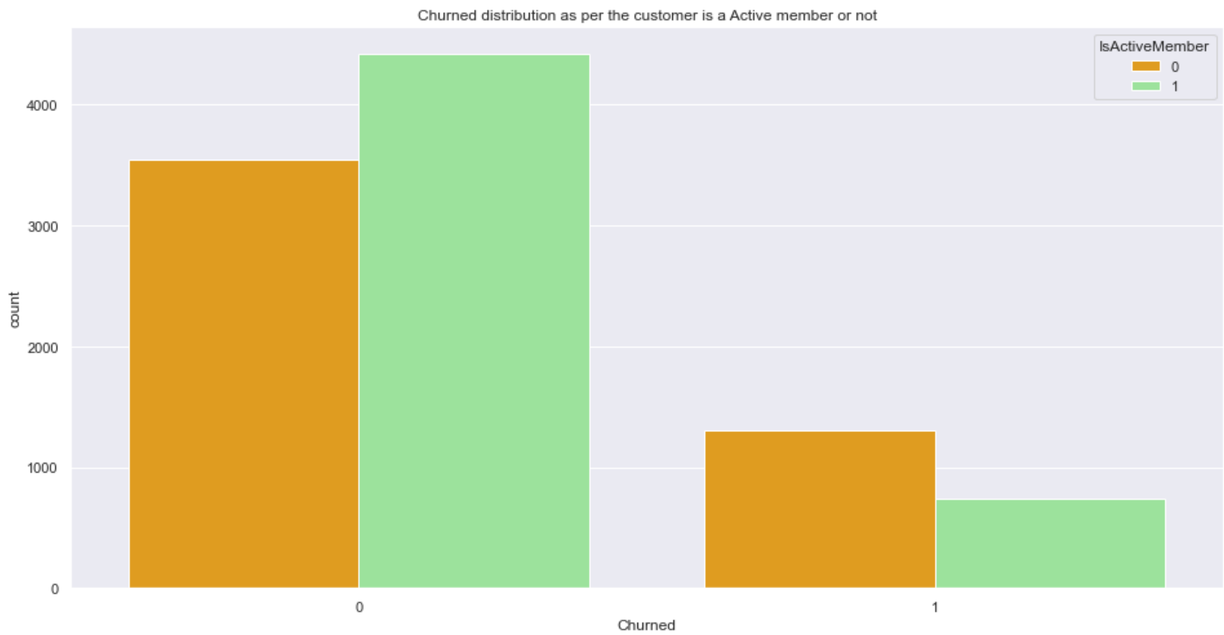


In [198]:

```
# 4) Churned distribution as per the customer is a Active member or not
```

In [199]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per the customer is a Active member or not")
sns.countplot(df['Churned'],hue='IsActiveMember',palette=['orange','lightgreen'],data=df);
```

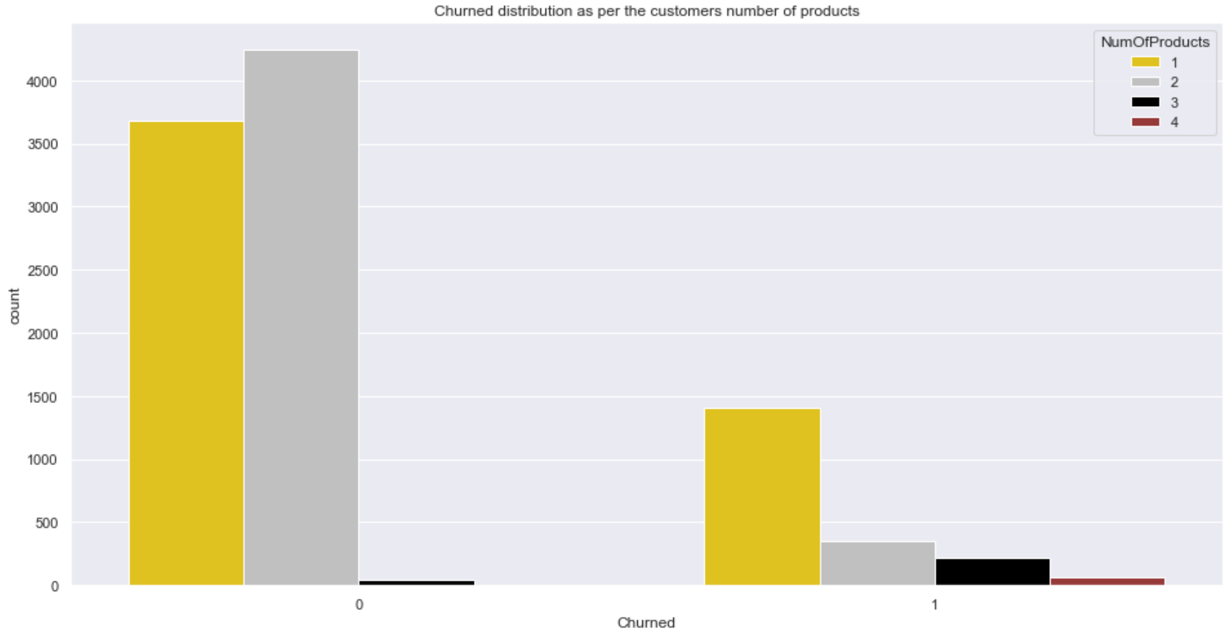


In [200]:

```
# 5)Churned distribution as per the customer NumOfProducts
```

In [201]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per the customers number of products")
sns.countplot(df['Churned'],hue='NumOfProducts',palette=['Gold','silver','black','brown'],data=df);
```

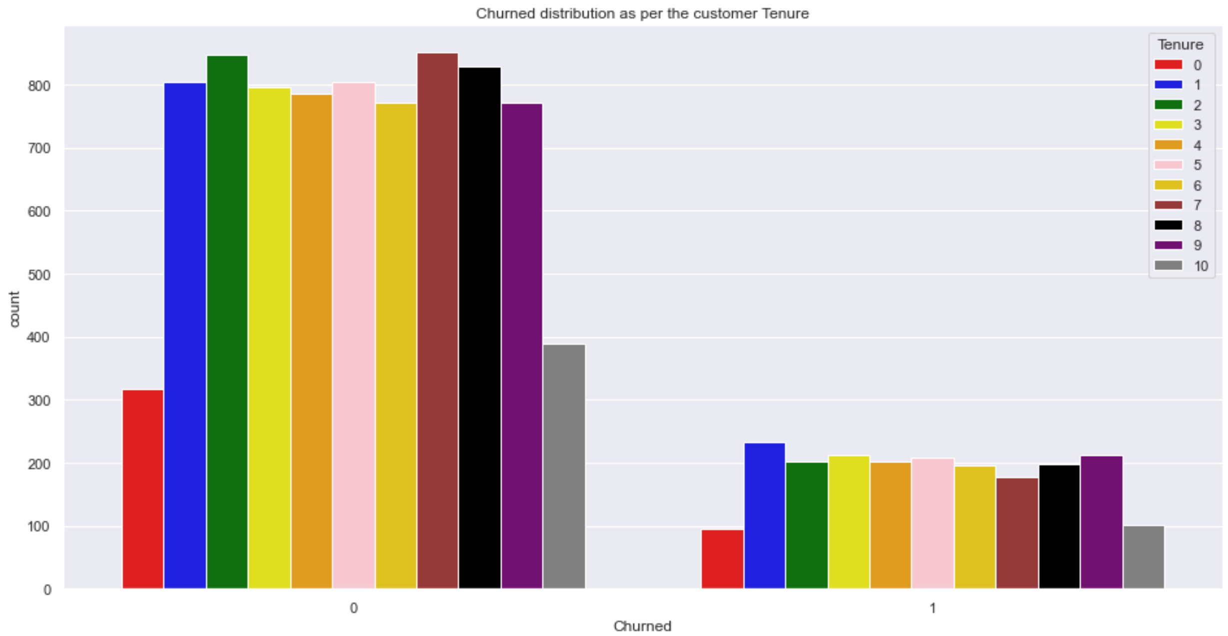


In [202]:

```
# 6) Churned distribution as per the customer Tenure
```

In [203]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per the customer Tenure")
sns.countplot(df['Churned'],hue='Tenure',palette=['red','blue','green','yellow','orange','pink','gold','brown','
```

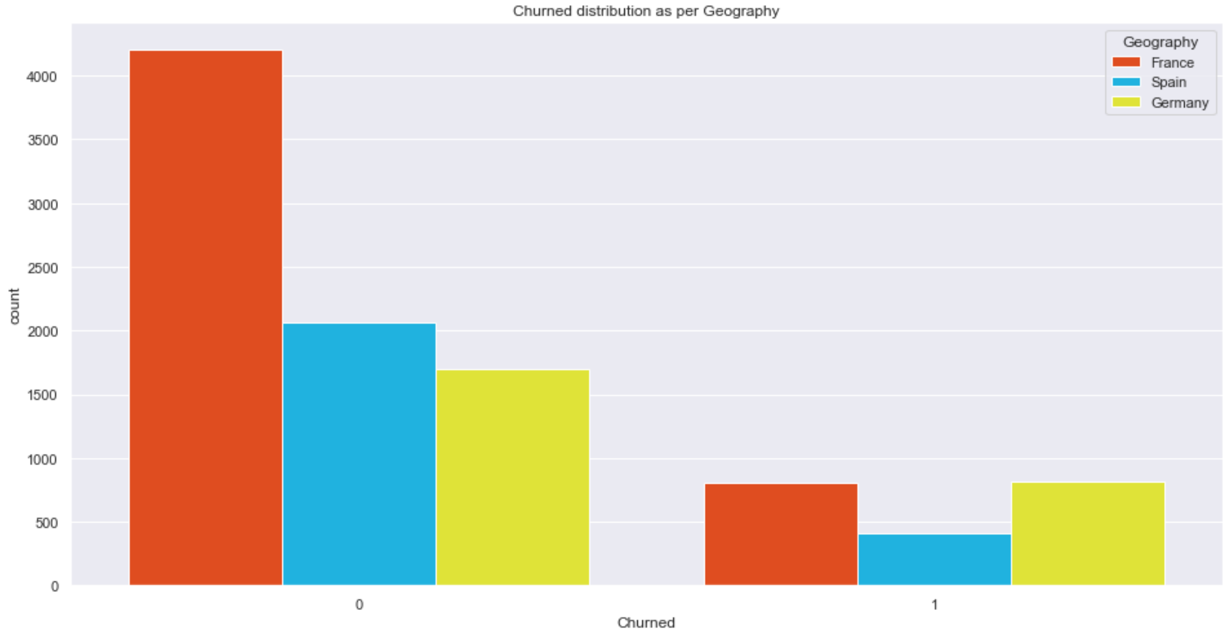


In [204]:

```
# 7) Churned distribution as per Geography
```

In [205]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per Geography")
sns.countplot(df['Churned'],hue='Geography',palette=['#FF3C00','#00C3FF','#FAFF1B'],data=df);
```

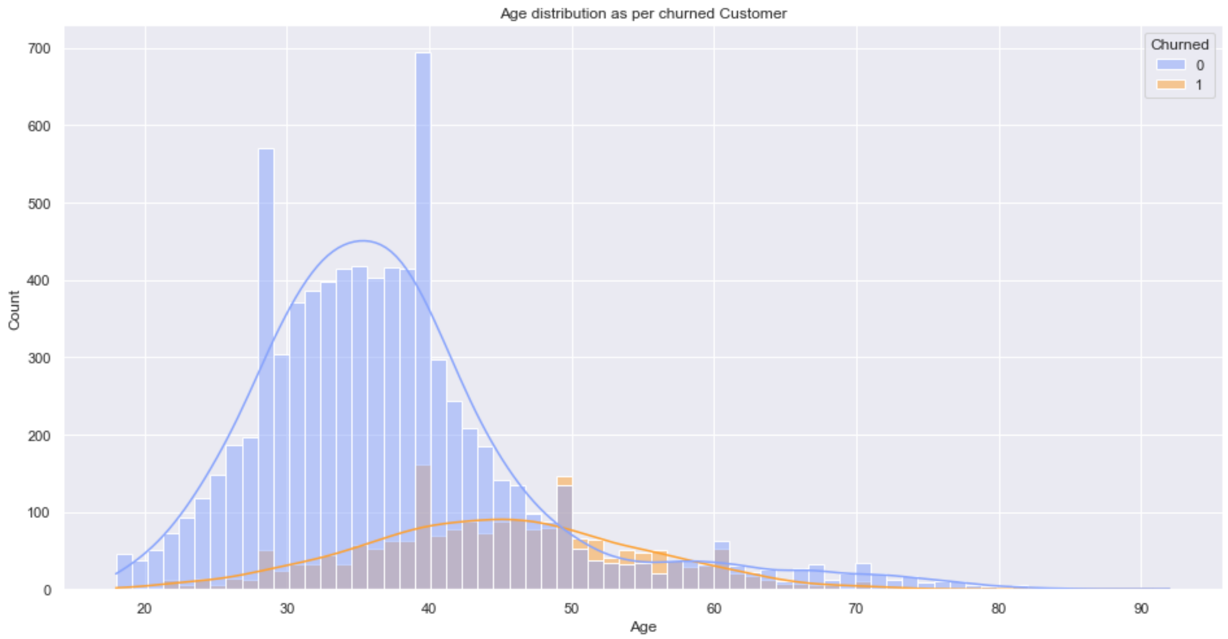


In [206]:

```
# 8) Age distribution as per churned Customer
```

In [207]:

```
plt.figure(figsize=(16,8))
plt.title("Age distribution as per churned Customer")
sns.histplot(x='Age', hue='Churned',palette=['#88A4FD','#FFA131'],kde=True,data=df);
```



In [208]:

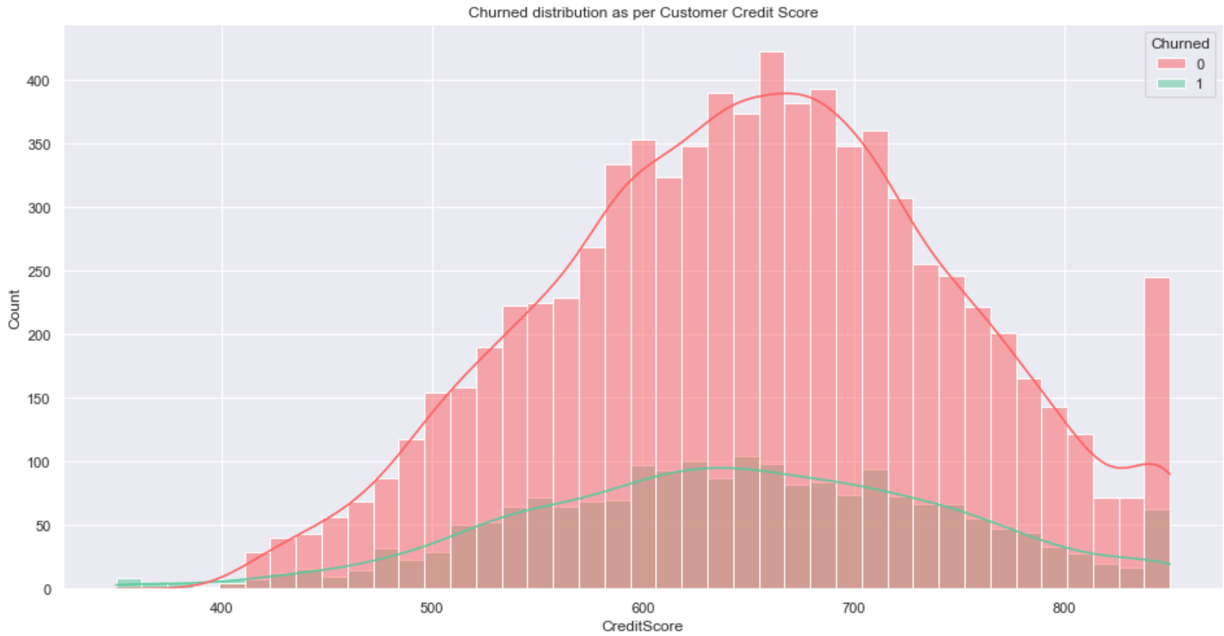
```
# 9)Churned distribution as per Customer Credit Score
```

In [209]:

```
plt.figure(figsize=(16,8))
plt.title("Churned distribution as per Customer Credit Score")
sns.histplot(x='CreditScore',hue=df['Churned'],palette=['#FF6061','#59CA9D'],kde=True,data=df)
```

Out[209]:

```
<AxesSubplot:title={'center':'Churned distribution as per Customer Credit Score'}, xlabel='CreditS
core', ylabel='Count'>
```



In [210]:

```
# 10) Churned distribution as per Customer Estimated salary
```

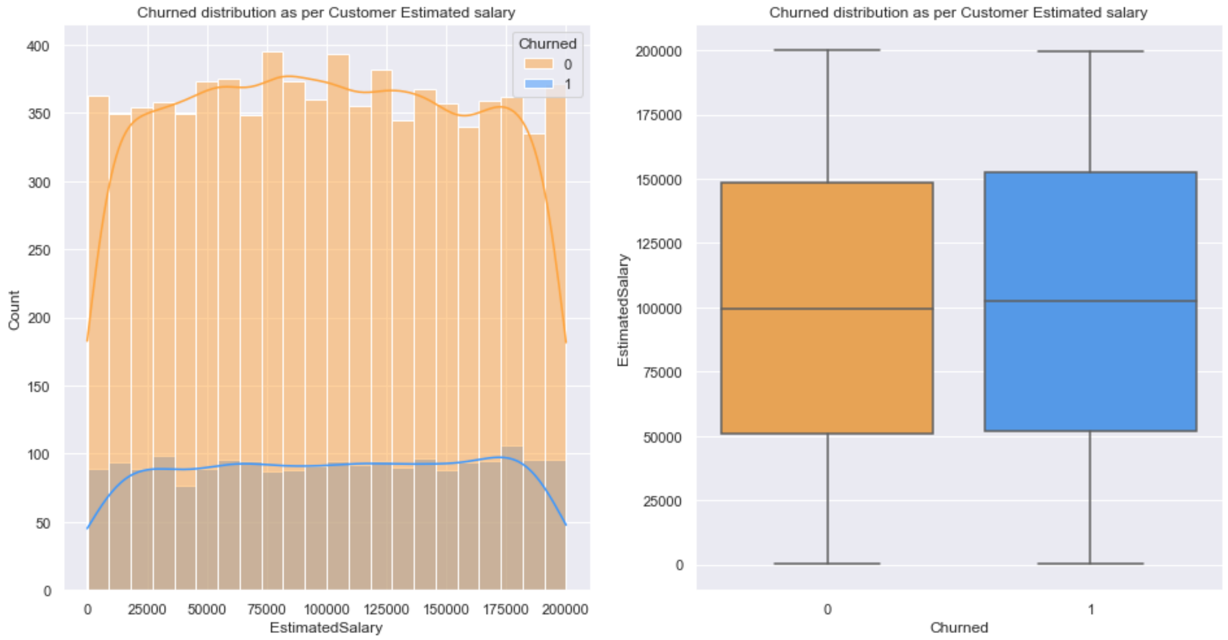
In [211]:

```
plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
plt.title("Churned distribution as per Customer Estimated salary")
sns.histplot(x='EstimatedSalary',hue=df['Churned'],palette=['#FFA43D','#3D98FF'],kde=True,data=df)

plt.subplot(1,2,2)
plt.title("Churned distribution as per Customer Estimated salary")
sns.boxplot(df['Churned'],df['EstimatedSalary'],palette=['#FFA43D','#3D98FF'],data=df)
```

Out[211]:

```
<AxesSubplot:title={'center':'Churned distribution as per Customer Estimated salary'}, xlabel='Chu
rned', ylabel='EstimatedSalary'>
```



In [212]:

```
df.head()
```

Out[212]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

In [213]:

```
# 11) Churned distribution as per Customer Bank Balance
```

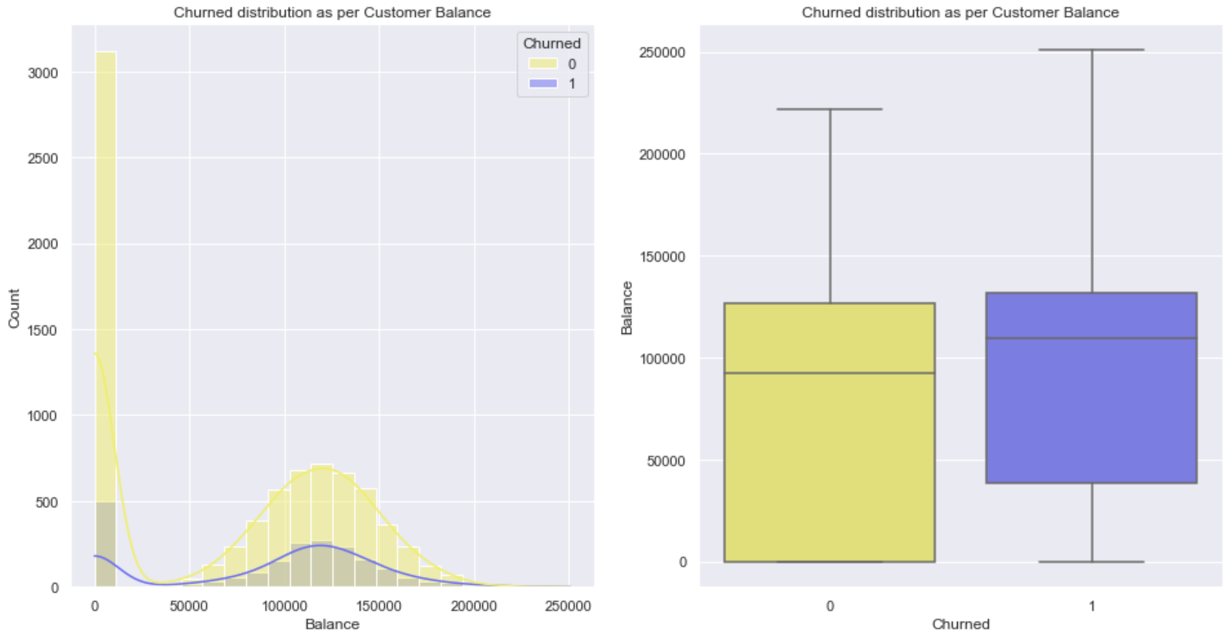
In [214]:

```
plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
plt.title("Churned distribution as per Customer Balance")
sns.histplot(x='Balance',hue=df['Churned'],palette=['#F1EF6B','#6B6DF1'],kde=True,data=df)

plt.subplot(1,2,2)
plt.title("Churned distribution as per Customer Balance")
sns.boxplot(df['Churned'],df['Balance'],palette=['#F1EF6B','#6B6DF1'],data=df)
```

Out[214]:

```
<AxesSubplot:title={'center':'Churned distribution as per Customer Balance'}, xlabel='Churned', yl
abel='Balance'>
```



NOW APPLY ONE HOT ENCODING ON CATEGORICAL COLUMNS

In [215]:

```
df=pd.get_dummies(columns=["Geography","Gender","NumOfProducts"],data=df)
```

In [216]:

```
df
```

Out[216]:

	CreditScore	Age	Tenure	Balance	HasCrCard	IsActiveMember	EstimatedSalary	Churned	Geography_France	Ge
0	619	42	2	0.00	1	1	101348.88	1	1	
1	608	41	1	83807.86	0	1	112542.58	0	0	
2	502	42	8	159660.80	1	0	113931.57	1	1	
3	699	39	1	0.00	0	0	93826.63	0	1	
4	850	43	2	125510.82	1	1	79084.10	0	0	
...	...	...	...	...	...	...	...	...	...	
9995	771	39	5	0.00	1	0	96270.64	0	1	
9996	516	35	10	57369.61	1	1	101699.77	0	1	
9997	709	36	7	0.00	0	1	42085.58	1	1	
9998	772	42	3	75075.31	1	0	92888.52	1	0	
9999	792	28	4	130142.79	1	0	38190.78	0	1	

10000 rows × 17 columns

Checking Skewness of Continous Features

In [222]:

```
print("The skewness of creditscore is :",df['CreditScore'].skew())
print("The skewness of Age is :",df['Age'].skew())
print("The skewness of EstimatedSalary is :",df['EstimatedSalary'].skew())
```

```
The skewness of creditscore is : -0.07160660820092675
The skewness of Age is : 1.0113202630234552
The skewness of EstimatedSalary is : 0.0020853576615585162
```

AS you clearly see that Age columns skewness is very high and highly right-skewed which result in that it contain high amount of outliers.

so we transform Age feature

Performing Log Transformation on Age Column.

In [223]:

```
old_age = df["Age"] # store Age feature in another variable for comparing it after transformation
```

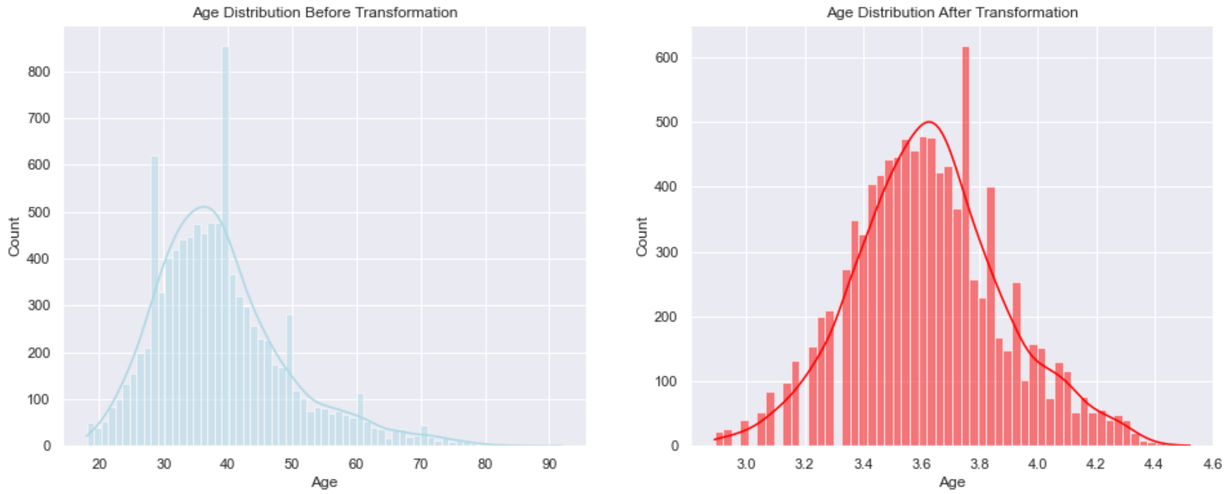
In [224]:

```
df["Age"] = np.log(df["Age"])  # use log transformation for creation for log normal distribution for age columns
```

In [227]:

```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.histplot(old_age, color="lightblue", kde=True)
plt.title("Age Distribution Before Transformation")

plt.subplot(1,2,2)
sns.histplot(df["Age"], color="red", kde=True)
plt.title("Age Distribution After Transformation");
```



Model Building

Segregating Features & Labels for Model Training

In [228]:

```
X = df.drop(columns=["Churned"])
y = df["Churned"]
```

Splitting Data For Model Training & Testing.

In [229]:

```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

In [231]:

```
print("Shape of x_train is:",x_train.shape)
print("Shape of x_test is: ",x_test.shape)
print("Shape of y_train is:",y_train.shape)
print("Shape of y_test is: ",y_test.shape)
```

```
Shape of x_train is: (8000, 16)
Shape of x_test is:  (2000, 16)
Shape of y_train is: (8000,)
Shape of y_test is:  (2000,)
```

In [232]:

```
dt=DecisionTreeClassifier()
```

In [233]:

```
dt.fit(x_train,y_train)
```

Out[233]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [234]:

```
y_pred= dt.predict(x_test)
```

In [237]:

```
y_train_pred=dt.predict(x_train)
```

In [238]:

```
accuracy_score(y_test,y_pred)
```

Out[238]:

```
0.78
```

In [241]:

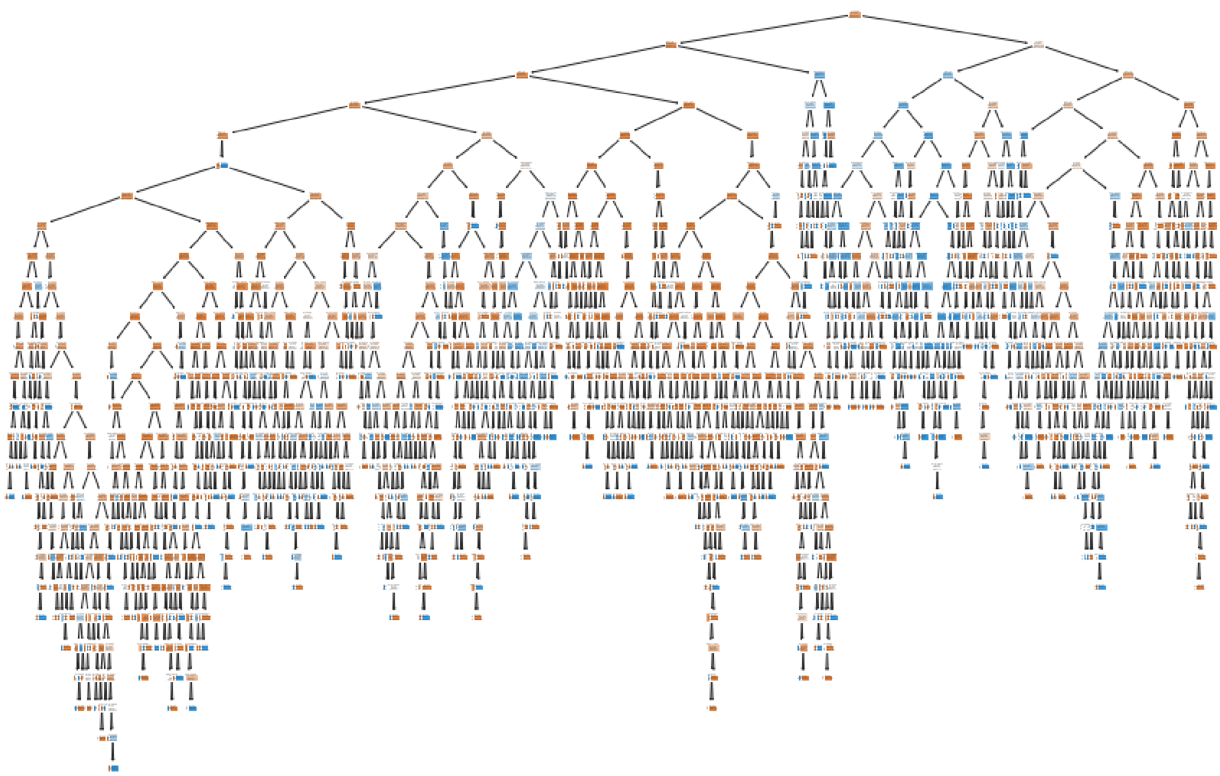
```
accuracy_score(y_train,y_train_pred)
```

Out[241]:

```
1.0
```

In [243]:

```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(dt,filled=True)
plt.show()
```



overfitting rescue by haperparameter tunning

In [244]:

```
grid_param ={
    'criterion':['gini','entropy'],
    'max_depth': range(2,32,1),
    'min_samples_leaf': range(1,10,1),
    'min_samples_split': range(2,10,1),
    'splitter':['best','random']
}
```

In [245]:

```
grid_search= GridSearchCV(estimator=dt,
                         param_grid= grid_param,
                         cv=5,
                         n_jobs=-1)
```

In [246]:

```
grid_search.fit(x_train,y_train)
```

Out[246]:

```
▸          GridSearchCV
▸ estimator: DecisionTreeClassifier
      ▸ DecisionTreeClassifier
```

In [248]:

```
best_parameters= grid_search.best_params_
print(best_parameters)
```

```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 7, 'min_samples_split': 9, 'splitte
r': 'random'}
```

In [249]:

```
grid_search.best_score_
```

Out[249]:

```
0.8561249999999999
```

In [250]:

```
dt= DecisionTreeClassifier(**best_parameters)
dt.fit(x_train,y_train)
```

Out[250]:

```
▾                        DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=7,
                       min_samples_split=9, splitter='random')
```

In [252]:

```
dt.score(x_test,y_test)
```

Out[252]:

```
0.861
```

In [253]:

```
y_pred= dt.predict(x_test)
```

In [255]:

```
y_pred_train=dt.predict(x_train)
```

Model Evaluation using Accuracy_score

In [268]:

```
accuracy_score(y_pred,y_test)
```

Out[268]:

```
0.861
```

In [269]:

```
accuracy_score(y_pred_train,y_train)
```

Out[269]:

```
0.85475
```

Model Evaluation using Different Metric Values

In [277]:

```
print("F1 Score of the Model is :",f1_score(y_pred,y_test,average="micro"))
print("Recall Score of the Model is :",recall_score(y_pred,y_test,average="micro"))
print("Precision Score of the Model is :",precision_score(y_pred,y_test,average="micro"))
```

```
F1 Score of the Model is : 0.861
Recall Score of the Model is : 0.861
Precision Score of the Model is : 0.861
```

Checking importance of each features

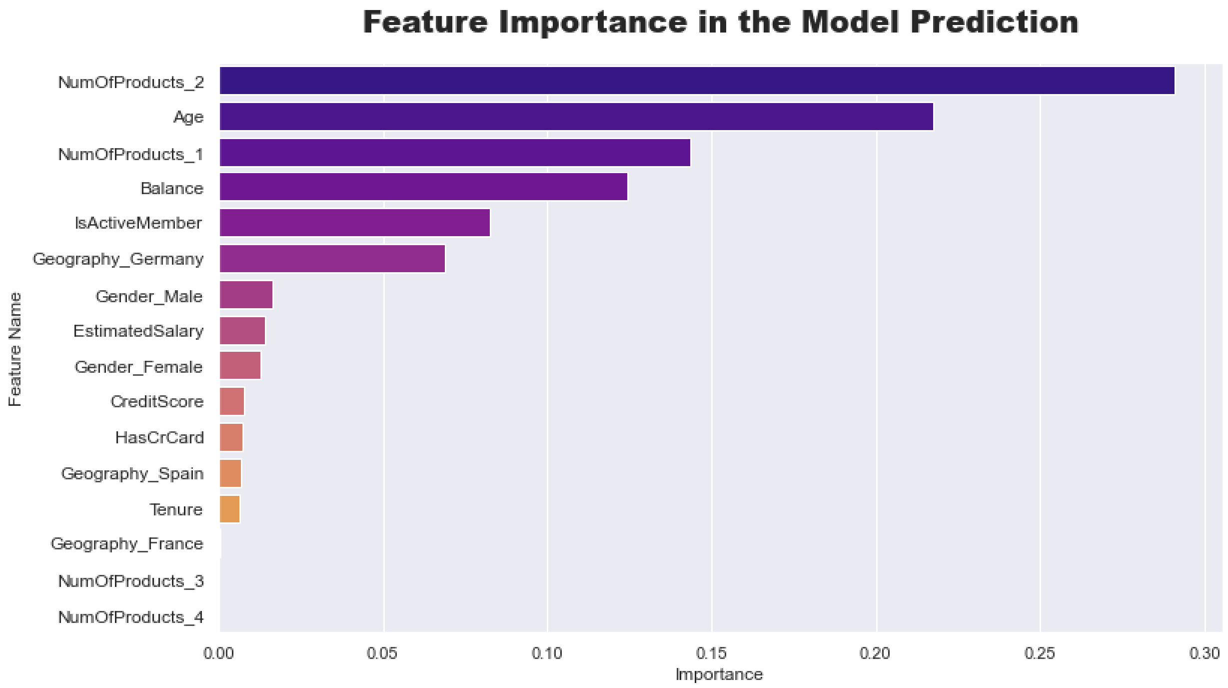
In [270]:

```
imp_df = pd.DataFrame({"Feature Name":x_train.columns,"Importance":dt.feature_importances_})
```

In [271]:

```
features = imp_df.sort_values(by="Importance",ascending=False)

plt.figure(figsize=(12,7))
sns.barplot(x="Importance", y="Feature Name", data=features, palette="plasma")
plt.title("Feature Importance in the Model Prediction", fontweight="black", size=20, pad=20)
plt.yticks(size=12)
plt.show()
```



Conculsion

The key factors that significantly influence the deactivation of customers banking facilities are Total_Products, Age, IsActiveMember, Gender and Geography.

High Training and Testing Accuracies: Both the model achieved a high accuracy score near to 90% on the training data, indicating a good fit to the training instances. Additionally, the model's accuracy score near to 85% on the testing data suggests its ability to generalize well to unseen instances.

High F1 Score, Recall, and Precision: The model achieved high F1 score, recall, and precision values, all approximately 0.8. This indicates that the model has a strong ability to correctly identify positive cases while minimizing false positives and maximizing true positives.

Future updation:

The bank can try to convince the customers to have atleast 2 banking products but not less than 2. The bank can launch a scheme for customers with higher ages (Senior Citizens) so that they not deactivate their banking facilities. The bank can provide Rewards and Incentive Programs, Regular Communication and Updates, and Enhanced Digital Services so that customers remain active to the banking facilities.