In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split,KFold,cross_val_score,GridSearchCV
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_curve

from io import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
#import pydotplus
```

In [2]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```python
#importing data
retail_customer=pd.read_csv('Retail_Customer_Insights.csv')
retail_customer.head()
```

Out[3]:

	Customer_ID	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital_Status	Number_of_Children	Employment_Status	Credit_Score	Owns_House	Monthly_Expenditure	Internet_Usage_Hours_per_Week
0	CID770487	45	72633.53	Non-binary	0	Electronics	9.0	541.11	Divorced	1	Unemployed	664	False	2159.63	33
1	CID216739	38	61816.55	Non-binary	0	Books	6.0	497.41	Married	3	Unemployed	623	True	1439.88	10
2	CID126225	47	57338.15	Non-binary	0	Grocery	3.0	634.90	Single	1	Student	342	True	1786.38	51
3	CID877572	58	83800.37	Female	0	Furniture	4.0	505.82	Divorced	2	Employed	810	True	NaN	45
4	CID388389	37	64875.12	Male	0	Furniture	6.0	610.39	Single	1	Student	692	True	1235.51	-9223372036854775808

In [4]:
```python
retail_customer.set_index('Customer_ID',inplace = True)
retail_customer.head()
```

Out[4]:

Customer_ID	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital_Status	Number_of_Children	Employment_Status	Credit_Score	Owns_House	Monthly_Expenditure	Internet_Usage_Hours_per_Week
CID770487	45	72633.53	Non-binary	0	Electronics	9.0	541.11	Divorced	1	Unemployed	664	False	2159.63	33
CID216739	38	61816.55	Non-binary	0	Books	6.0	497.41	Married	3	Unemployed	623	True	1439.88	10
CID126225	47	57338.15	Non-binary	0	Grocery	3.0	634.90	Single	1	Student	342	True	1786.38	51
CID877572	58	83800.37	Female	0	Furniture	4.0	505.82	Divorced	2	Employed	810	True	NaN	45
CID388389	37	64875.12	Male	0	Furniture	6.0	610.39	Single	1	Student	692	True	1235.51	-9223372036854775808

In [5]:
```python
retail_customer['Purchase_History'].value_counts()
```

Out[5]:
```
Purchase_History
0    59970
1    40030
Name: count, dtype: int64
```

In [6]:
```python
retail_customer.isnull().sum()
```

Out[6]:
```
Age                                 0
Annual_Income                    5000
Gender                              0
Purchase_History                    0
Product_Category                    0
Customer_Satisfaction            3000
Loyalty_Points                   2000
Marital_Status                      0
Number_of_Children                  0
Employment_Status                   0
Credit_Score                        0
Owns_House                          0
Monthly_Expenditure              5000
Internet_Usage_Hours_per_Week       0
dtype: int64
```

In [7]:
```python
retail_customer.isna().sum()
```

Out[7]:
```
Age                                 0
Annual_Income                    5000
Gender                              0
Purchase_History                    0
Product_Category                    0
Customer_Satisfaction            3000
Loyalty_Points                   2000
Marital_Status                      0
Number_of_Children                  0
Employment_Status                   0
Credit_Score                        0
Owns_House                          0
Monthly_Expenditure              5000
Internet_Usage_Hours_per_Week       0
dtype: int64
```

In [8]:
```python
#removing null/nan values

retail_customer.fillna({'Annual_Income':retail_customer['Annual_Income'].mean(),'Customer_Satisfaction':retail_customer['Customer_Satisfaction'].mean(),'Loyalty_Points':retail_customer['Loyalty_Points'].mean(),'Monthly_Expenditure':retail_customer['Monthly_Expenditure'].mean()},inplace=True)
```

In [9]:
```python
retail_customer.isnull().sum()
```

Out[9]:
```
Age                              0
Annual_Income                    0
Gender                           0
Purchase_History                 0
Product_Category                 0
Customer_Satisfaction            0
Loyalty_Points                   0
Marital_Status                   0
Number_of_Children               0
Employment_Status                0
Credit_Score                     0
Owns_House                       0
Monthly_Expenditure              0
Internet_Usage_Hours_per_Week    0
dtype: int64
```

In [10]:
```python
retail_customer.isna().sum()
```

Out[10]:
```
Age                              0
Annual_Income                    0
Gender                           0
Purchase_History                 0
Product_Category                 0
Customer_Satisfaction            0
Loyalty_Points                   0
Marital_Status                   0
Number_of_Children               0
Employment_Status                0
Credit_Score                     0
Owns_House                       0
Monthly_Expenditure              0
Internet_Usage_Hours_per_Week    0
dtype: int64
```

In [11]:
```python
# Removing outliers from Age column
dataset=np.array(retail_customer['Age']).tolist()
dataset.sort()
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=retail_customer.loc[(retail_customer['Age']>ll) & (retail_customer['Age']<ul)]
```

In [12]:
```python
# Removing outliers from Annual_Income column
dataset=np.array(data['Annual_Income']).tolist()
dataset.sort()
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Annual_Income']>ll) & (data['Annual_Income']<ul)]
```

In [13]:
```python
# Removing outliers from Loyalty_Points column
dataset=np.array(data['Loyalty_Points']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Loyalty_Points']>ll) & (data['Loyalty_Points']<ul)]
```

In [14]:
```python
# Removing outliers from Number_of_Children column
dataset=np.array(data['Number_of_Children']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Number_of_Children']>ll) & (data['Number_of_Children']<ul)]
```

In [15]:
```python
# Removing outliers from Credit_Score column
dataset=np.array(data['Credit_Score']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Credit_Score']>ll) & (data['Credit_Score']<ul)]
```

In [16]:
```python
# Removing outliers from Number_of_Children column
dataset=np.array(data['Monthly_Expenditure']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Monthly_Expenditure']>ll) & (data['Monthly_Expenditure']<ul)]
```

In [17]:
```python
# Removing outliers from Internet_Usage_Hours_per_Week column
dataset=np.array(data['Internet_Usage_Hours_per_Week']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Internet_Usage_Hours_per_Week']>ll) & (data['Internet_Usage_Hours_per_Week']<ul)]
```

In [18]:
```python
'''for col in data.columns:
    if col!='Customer_ID':
        plt.figure()
        sns.boxplot(data[col],orient='h')'''
```

Out[18]:
```
"for col in data.columns:\n    if col!='Customer_ID':\n        plt.figure()\n        sns.boxplot(data[col],orient='h')"
```

In [19]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 81315 entries, CID770487 to CID966793
Data columns (total 14 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Age                            81315 non-null  int64
 1   Annual_Income                  81315 non-null  float64
 2   Gender                         81315 non-null  object
 3   Purchase_History               81315 non-null  int64
 4   Product_Category               81315 non-null  object
 5   Customer_Satisfaction          81315 non-null  float64
 6   Loyalty_Points                 81315 non-null  float64
 7   Marital_Status                 81315 non-null  object
 8   Number_of_Children             81315 non-null  int64
 9   Employment_Status              81315 non-null  object
 10  Credit_Score                   81315 non-null  int64
 11  Owns_House                     81315 non-null  bool
 12  Monthly_Expenditure            81315 non-null  float64
 13  Internet_Usage_Hours_per_Week  81315 non-null  int64
dtypes: bool(1), float64(4), int64(5), object(4)
memory usage: 8.8+ MB
```

In [20]:
```python
cat_col = ['Gender','Product_Category','Marital_Status','Employment_Status','Owns_House']
for cols in cat_col:
    le = LabelEncoder()
    data[cols]=le.fit_transform(data[cols])
```

In [21]:
```python
data.head()
```

Out[21]:

Customer_ID	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital_Status	Number_of_Children	Employment_Status	Credit_Score	Owns_House	Monthly_Expenditure	Internet_Usage_Hours_per_Week
CID770487	45	72633.53	2	0	2	9.0	541.11	0	1	3	664	0	2159.630000	33
CID216739	38	61816.55	2	0	0	6.0	497.41	1	3	3	623	1	1439.880000	10
CID126225	47	57338.15	2	0	4	3.0	634.90	2	1	2	342	1	1786.380000	51
CID877572	58	83800.37	0	0	3	4.0	505.82	0	2	0	810	1	1997.943548	45
CID356787	37	57270.25	3	1	4	3.0	458.98	0	4	0	844	1	1522.760000	16

In [22]:
```python
x=data.iloc[:,data.columns!='Purchase_History']
y=data.iloc[:,data.columns=='Purchase_History']
```

In [23]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=75)
```

In [24]:
```python
#Feature Scaling
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [25]:
```python
fit_rf = RandomForestClassifier(random_state = 40)
```

In [26]:
```python
param_dict = {'max_depth': [2,3,4,5],
              'bootstrap': [True,False],
              'max_features': ['auto','log2','sqrt'],
              'criterion': ['gini','entropy']}
cv_rf = GridSearchCV(fit_rf,cv=10,param_grid=param_dict,n_jobs=3)
cv_rf.fit(x_train,y_train)
```

Out[26]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

In [27]:
```python
print(f'Best Parameters are : {cv_rf.best_params_}')
```

```
Best Parameters are : {'bootstrap': True, 'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto'}
```

In [28]:
```python
print(f'Best score is : {cv_rf.best_score_}')
```

```
Best score is : 0.597322143126678
```

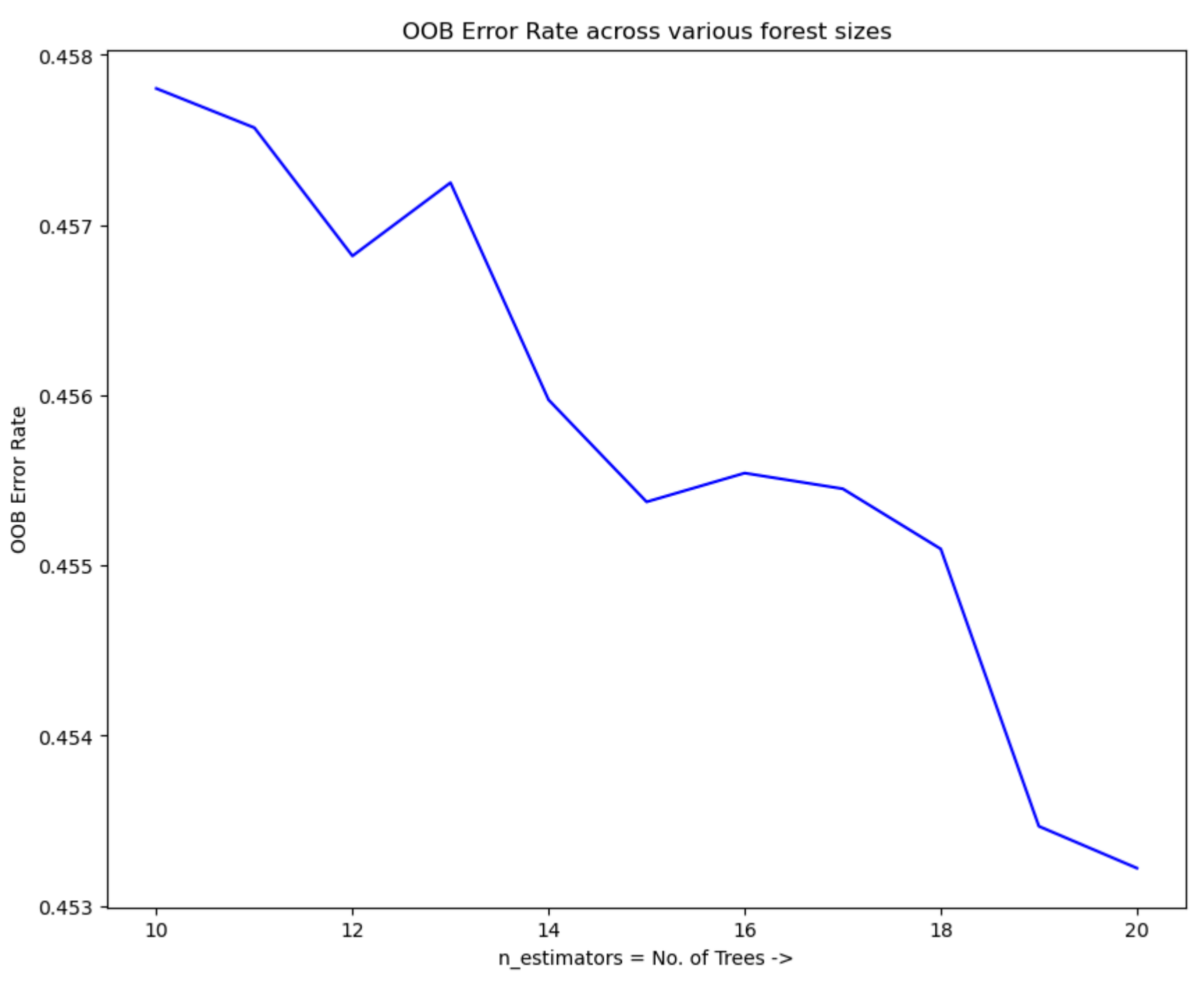
In [29]:
```python
fit_rf.set_params(warm_start=True,oob_score=True)
min_estimator=10
max_estimator=20
error_rate= {}
for i in range(min_estimator,max_estimator+1):
    fit_rf.set_params(n_estimators=i)
    fit_rf.fit(x_train,y_train)
    oob_error = 1- fit_rf.oob_score_
    error_rate[i]=oob_error
```

In [30]:
```python
oob_series = pd.Series(error_rate)
```

In [31]:
```python
fig, ax = plt.subplots(figsize=(10,8))

oob_series.plot(kind='line', color = 'blue')
plt.xlabel('n_estimators = No. of Trees ->')
plt.ylabel('OOB Error Rate')
plt.title('OOB Error Rate across various forest sizes')
```

Out[31]:
```
Text(0.5, 1.0, 'OOB Error Rate across various forest sizes')
```



In [35]:
```python
fit_rf.set_params(n_estimators = 16, bootstrap = True,criterion = 'entropy', max_depth= 8, max_features= 'auto', warm_start = False, oob_score= False)
```

Out[35]:
```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=8, max_features='auto',
                       n_estimators=16, random_state=40)
```

In [37]:
```python
fit_rf.fit(x_train_scaled, y_train)
```

Out[37]:
```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=8, max_features='auto',
                       n_estimators=16, random_state=40)
```

In [39]:
```python
train_accuracy = fit_rf.score(x_train_scaled, y_train)
print(train_accuracy)
```

```
0.5992283096599643
```

In [41]:
```python
test_accuracy = fit_rf.score(x_test_scaled, y_test)
print(test_accuracy)
```

```
0.6043780360327123
```

In [43]:
```python
importances = fit_rf.feature_importances_
importances
```

Out[43]:
```
array([0.10392011, 0.16187735, 0.02243501, 0.03661118, 0.04656097,
       0.151497  , 0.02810104, 0.03505016, 0.03081804, 0.11967929,
       0.0127999 , 0.14886701, 0.10178297])
```

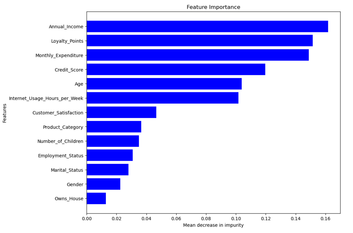
In [49]:
```python
feature_names= x_train.columns

feature_importance = pd.DataFrame({'Feature': feature_names, 'Importance':importances})
feature_importance = feature_importance.sort_values(by='Importance',ascending = False)
```

In [51]:
```python
# Plot the feature importance

plt.figure(figsize=(10,8))
plt.barh(feature_importance['Feature'],feature_importance['Importance'], color = 'blue')
plt.gca().invert_yaxis()
plt.xlabel('Mean decrease in impurity')
plt.ylabel('Features')
plt.title('Feature Importance')
```

Out[51]:
```
Text(0.5, 1.0, 'Feature Importance')
```


Feature Importance
Annual_Income
Loyalty_Points
Monthly_Expenditure
Credit_Score
Age
Internet_Usage_Hours_per_Week
Customer_Satisfaction
Product_Category
Number_of_Children
Employment_Status
Marital_Status
Gender
Owns_House
Features
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Mean decrease in impurity