- Solve the classification problem using the following methods: Logistic Regression, Decision Tree, Random Forest, XGBoost, Naïve Bayes, and SVM.
- Compare the performance of each classifier.
- Visualize the decision boundary for each method.

There is a dataset given which contains the information of various users obtained from the social networking sites. There is a car making company that has recently launched a new SUV car. So the company wanted to check how many users from the dataset, wants to purchase the car.

DataSet: Social_Network_Ads.csv

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd


df=pd.read_csv('/content/Social_Network_Ads.csv')
df.head()
df.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```
# seperate the attributes and target varibales from the dataframe
X = df[['Age',  'EstimatedSalary']].values
y = df['Purchased'].values


# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(X, y, test_size= 0.25, random_state=0)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

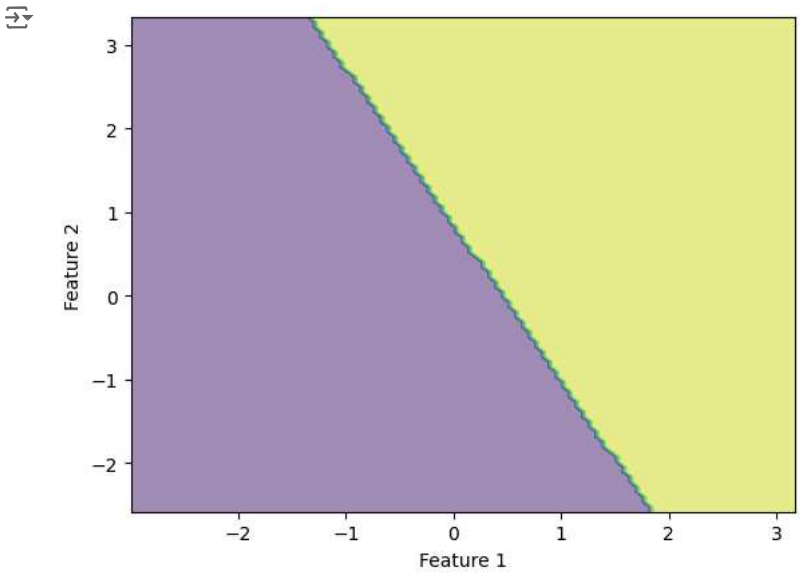
Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

logistic_classifier = LogisticRegression()
logistic_classifier.fit(x_train, y_train)

y_pred = logistic_classifier.predict(x_test)

print('Classification Report:')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.89      0.96      0.92        68
              1       0.89      0.75      0.81        32

       accuracy                           0.89       100
      macro avg       0.89      0.85      0.87       100
   weighted avg       0.89      0.89      0.89       100

   Confusion Matrix:
   [[65  3]
    [ 8 24]]
```

```
from sklearn.inspection import DecisionBoundaryDisplay

# sketch the decision boundary using DecisionBoundaryDisplay and the method from_estimator()
disp = DecisionBoundaryDisplay.from_estimator(logistic_classifier, x_train, response_method="predict",
                                              xlabel='Feature 1', ylabel='Feature 2', alpha=0.5)
# make a scatter plot of the data
#disp.ax_.scatter(x_train[:, 0], x_train[:, 1], c=y_train, edgecolor="k")
#plt.title('Logistic Regression (Training set)')
#plt.show()
```



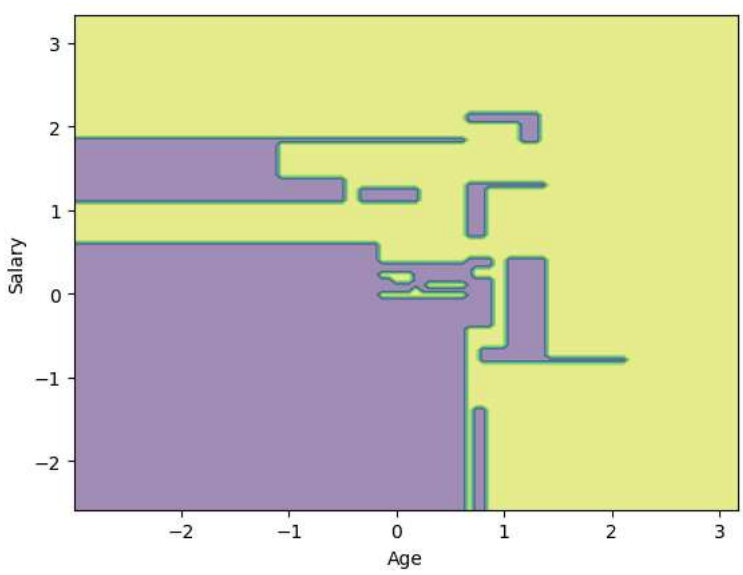
Descision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Fitting Decision Tree classifier to the training set
decision_tree_classifier = DecisionTreeClassifier(criterion='entropy', random_state=1)
decision_tree_classifier.fit(x_train, y_train)

# Predicting the test set results
y_pred = decision_tree_classifier.predict(x_test)

# Classification Report
print('Classification Report:')
print(classification_report(y_test, y_pred))

# Confusion Matrix
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        68
           1       0.83      0.91      0.87        32

    accuracy                           0.91       100
   macro avg       0.89      0.91      0.90       100
```

```
weighted avg       0.91      0.91      0.91       100

Confusion Matrix:
[[62  6]
 [ 3 29]]
```

```
from sklearn.inspection import DecisionBoundaryDisplay

# sketch the decision boundary using DecisionBoundaryDisplay and the method from_estimator()
disp = DecisionBoundaryDisplay.from_estimator(decision_tree_classifier, x_train, response_method="predict",
                                              xlabel='Age', ylabel='Salary', alpha=0.5,)

# make a scatter plot of the data
#disp.ax_.scatter(x[:, 0], x[:, 1], c=dataset.Purchased, edgecolor="k")
```

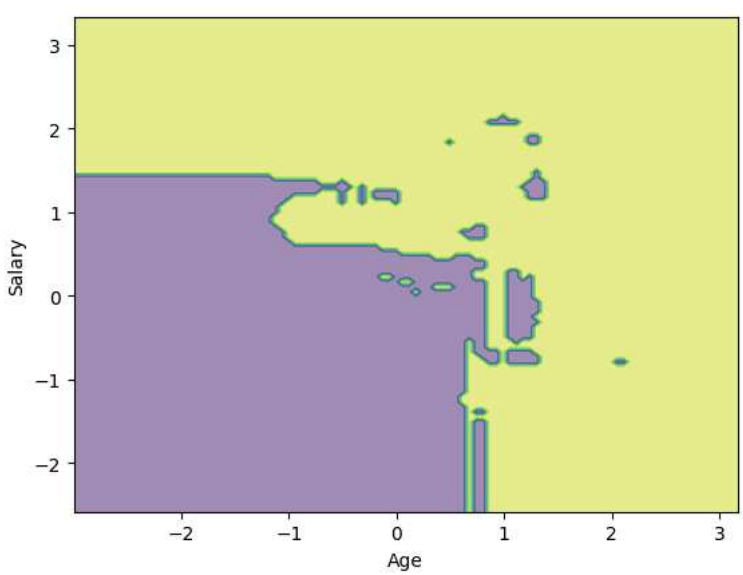


Random Forset

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

random_forest_classifier = RandomForestClassifier(n_estimators=100, random_state=1)
random_forest_classifier.fit(x_train, y_train)

y_pred = random_forest_classifier.predict(x_test)

print('Classification Report:')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        68
           1       0.88      0.88      0.88        32

    accuracy                           0.92       100
   macro avg       0.91      0.91      0.91       100
weighted avg       0.92      0.92      0.92       100

Confusion Matrix:
[[64  4]
 [ 4 28]]
```

```
from sklearn.inspection import DecisionBoundaryDisplay

# sketch the decision boundary using DecisionBoundaryDisplay and the method from_estimator()
disp = DecisionBoundaryDisplay.from_estimator(random_forest_classifier, x_train, response_method="predict",
                                              xlabel='Age', ylabel='Salary', alpha=0.5,)

# make a scatter plot of the data
# disp.ax_.scatter(x[:, 0], x[:, 1], c=dataset.Purchased, edgecolor="k")
```

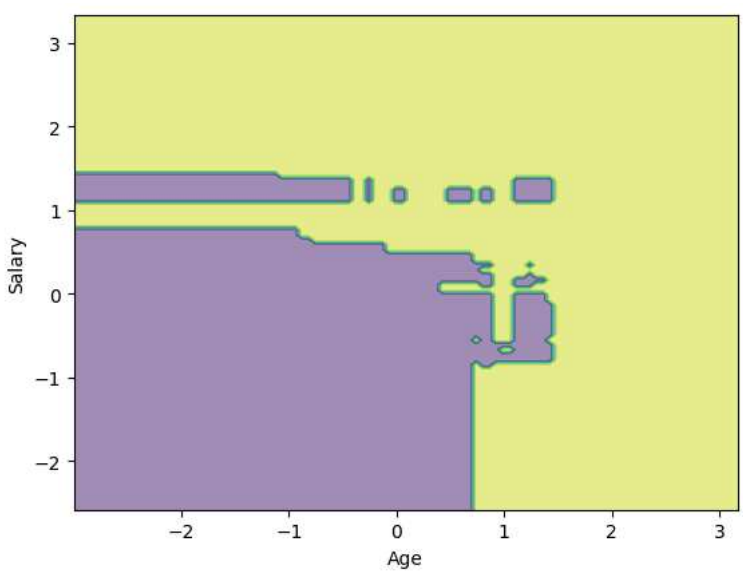


XGBoost

```
import xgboost as xgb
from sklearn.metrics import classification_report, confusion_matrix

xgb_classifier = xgb.XGBClassifier()
xgb_classifier.fit(x_train, y_train)

y_pred = xgb_classifier.predict(x_test)

print('Classification Report:')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93        68
           1       0.87      0.84      0.86        32

    accuracy                           0.91       100
   macro avg       0.90      0.89      0.90       100
weighted avg       0.91      0.91      0.91       100

Confusion Matrix:
[[64  4]
 [ 5 27]]
```

```
from sklearn.inspection import DecisionBoundaryDisplay

# sketch the decision boundary using DecisionBoundaryDisplay and the method from_estimator()
disp = DecisionBoundaryDisplay.from_estimator(xgb_classifier, x_train, response_method="predict",
                                              xlabel='Age', ylabel='Salary', alpha=0.5,)

# make a scatter plot of the data
# disp.ax_.scatter(x[:, 0], x[:, 1], c=dataset.Purchased, edgecolor="k")
```

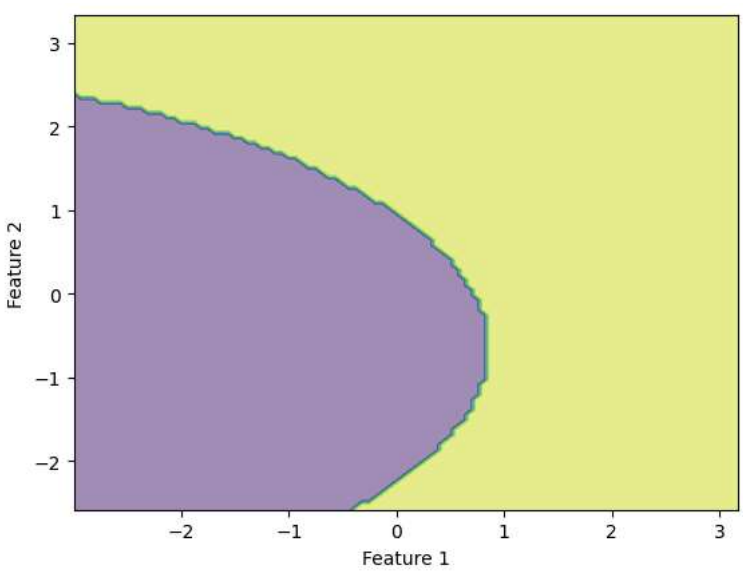


Naïve Bayes

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix

# Fitting Naïve Bayes classifier to the training set
naive_bayes_classifier = GaussianNB()
naive_bayes_classifier.fit(x_train, y_train)

# Predicting the test set results
y_pred = naive_bayes_classifier.predict(x_test)

# Classification Report
print('Classification Report:')
print(classification_report(y_test, y_pred))

# Confusion Matrix
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.96      0.93        68
           1       0.89      0.78      0.83        32

    accuracy                           0.90       100
   macro avg       0.90      0.87      0.88       100
weighted avg       0.90      0.90      0.90       100

Confusion Matrix:
[[65  3]
 [ 7 25]]
```

```
# Plotting decision boundary
disp = DecisionBoundaryDisplay.from_estimator(naive_bayes_classifier, x_train, response_method="predict",
                                              xlabel='Feature 1', ylabel='Feature 2', alpha=0.5)

# Make a scatter plot of the data
#disp.ax_.scatter(x_train[:, 0], x_train[:, 1], c=y_train, edgecolor="k")
#plt.title('Naive Bayes (Training set)')
#plt.show()
```



SVM

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Fitting SVM classifier to the training set
svm_classifier = SVC(kernel='linear', random_state=1)
svm_classifier.fit(x_train, y_train)

# Predicting the test set results
y_pred = svm_classifier.predict(x_test)

# Classification Report
print('Classification Report:')
print(classification_report(y_test, y_pred))

# Confusion Matrix
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.97      0.93        68
           1       0.92      0.75      0.83        32

    accuracy                           0.90       100
   macro avg       0.91      0.86      0.88       100
weighted avg       0.90      0.90      0.90       100

Confusion Matrix:
[[66  2]
 [ 8 24]]
```

```
from sklearn.inspection import DecisionBoundaryDisplay

# sketch the decision boundary using DecisionBoundaryDisplay and the method from_estimator()
disp = DecisionBoundaryDisplay.from_estimator(random_forest_classifier, x_train, response_method="predict",
                                              xlabel='Age', ylabel='Salary', alpha=0.5,)

# make a scatter plot of the data
# disp.ax_.scatter(x[:, 0], x[:, 1], c=dataset.Purchased, edgecolor="k")
```

[4]