Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2021-2022)
EC344 - Machine Learning and AI

Experiment 5 : Implement different Classifier

Name: Abhinav Pallayil **Roll No.:** 2019120045 **Date:** 07-11-2022

Objective: To explore the different classifier on different dataset

Outcomes:

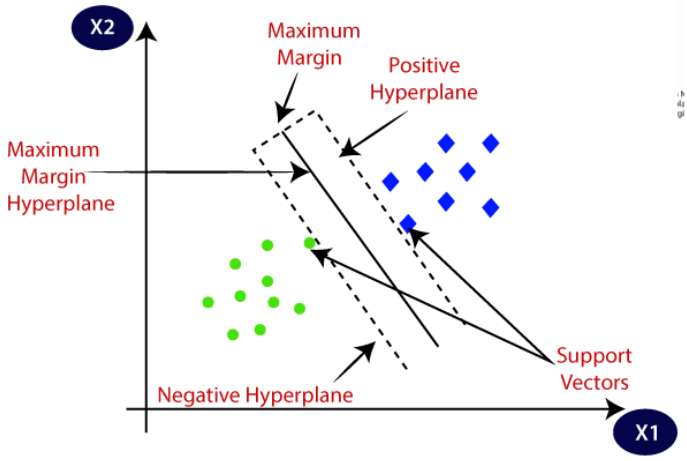
1. Identifying the classifier based on the dataset
2. Build the model and classify it using linear and nonlinear classifier(SVM , SVR, Random forest, KNN)
3. Draw various plots and interpret them

System Requirements:

Linux OS with Python and libraries or R or windows with MATLAB

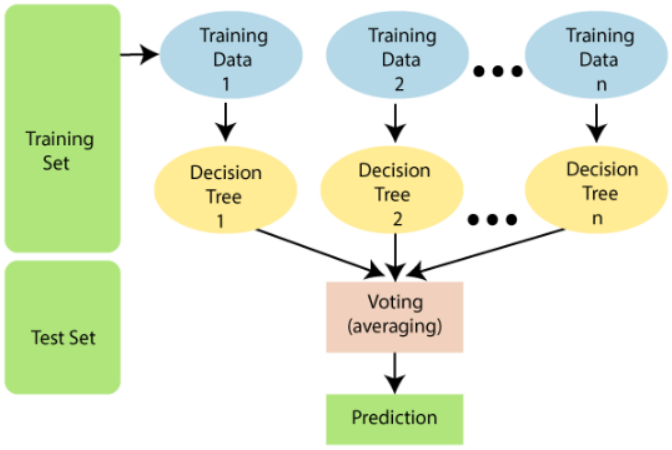
Theory:

1. **SVM**
    a. Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
    b. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.



c. *Linear SVM:* Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
d. *Non-linear SVM:* Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

2. **SVR**
   a. SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data.
   b. SVR is a powerful algorithm that allows us to choose how tolerant we are of errors, both through an acceptable error margin(ϵ) and through tuning our tolerance of falling outside that acceptable error rate.
   c. The model produced by support vector classification (as described above) depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin
   d. The model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction
3. **Random forest**
   a. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.



   b. Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

c. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
d. It takes less training time as compared to other algorithms. It predicts output with high accuracy, even for the large dataset it runs efficiently. It can also maintain accuracy when a large proportion of data is missing.
e. There are mainly four sectors where Random forest mostly used:
Banking for the identification of loan risk, Medicine for disease trends and risks of the disease can be identified, Land Use for identifying the areas of similar land use by this algorithm, Marketing for identifying trends using this algorithm.

4. **KNN**
    a. K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
    b. The K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.
    c. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.
    d. The K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
    e. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
    f. The KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
    g. It is simple to implement, It is robust to the noisy training data, It can be more effective if the training data is large.

Dataset Description:

1. **SVM**
    Dataset describes the purchases made by the employee and their estimated salary based on it.
    a. User ID : User Identifier by id
    b. Gender : Gender of the user
    c. Age : Age of the user
    d. EstimatedSalary: Estimated salary of the user
    e. Purchased : How many purchases have been made under that employee

2. **SVR**

   Dataset describes the salaries of the user based on their designation and level.

   a. Position : Designation/Position of the user
   b. Level : Level of the designation of user
   c. Salary : Salary of the user

3. **Random Forest**

   Dataset describes the purchases made by the employee and their estimated salary based on it.

   a. User ID : User Identifier by id
   b. Gender : Gender of the user
   c. Age : Age of the user
   d. EstimatedSalary: Estimated salary of the user
   e. Purchased : How many purchases have been made under that employee

4. **KNN**

   Dataset describes the purchases made by the employee and their estimated salary based on it.

   a. Age : Age of the user
   b. EstimatedSalary : Estimated salary of the user
   c. Purchased : Purchases made by the user

Code:

1. **SVM**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.colors import ListedColormap
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

dataset = pd.read_csv('Social_Network_Ads.csv')
dataset.info()
dataset.head()

X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.25, random_state = 0)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracy_score(y_test,y_pred)

X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

2. **SVR**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
```

```
dataset = pd.read_csv('Position_Salaries.csv')

dataset.info()
dataset.head()

x = dataset.iloc[:,1:2].values
y = dataset.iloc[:,-1].values
x = x.reshape(-1,1)
y = y.reshape(-1,1)

standardscaler_x = StandardScaler()
x = standardscaler_x.fit_transform(x)
standardscaler_y = StandardScaler()
y = standardscaler_y.fit_transform(y)

y = y.reshape(len(y),)

regressor = SVR(kernel='poly')
regressor = regressor.fit(x,y)

test = np.zeros(1) # we are testing just one value
test[0]= 6.5
test = test.reshape(1,1) # reshape to 2D array!
test = standardscaler_x.transform(test) # rescaling test data like
train data

y_pred = regressor.predict(test)
y_pred = y_pred.reshape(-1,1)

y_predict = standardscaler_y.inverse_transform(y_pred)
plt.scatter(x,y, color ='red', alpha=0.6)
plt.scatter(test,y_pred,color = 'blue', marker='D')
plt.plot(x,regressor.predict(x),color='green')
plt.title('Level vs Salary (train data) using poly kernel')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.legend()
plt.grid()
plt.show()
```

```
regressor2 = SVR(kernel='rbf')
regressor2 = regressor2.fit(x,y)

y_pred = regressor2.predict(test)
y_pred = y_pred.reshape(-1,1)

y_predict = standardscaler_y.inverse_transform(y_pred)

plt.scatter(x,y, color ='red', alpha=0.6)
plt.scatter(test,y_pred,color = 'blue', marker='D')
plt.plot(x,regressor2.predict(x),color='green')
plt.title('Level vs Salary (train data) using rbf kernel')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.legend()
plt.grid()
plt.show()
```

3. Random Forest

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

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)

print(X_train)
print(y_train)
print(X_test)
print(y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)

print(X_train)
print(X_test)

from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion =
'entropy', random_state = 0)
classifier.fit(X_train, y_train)

print(classifier.predict(sc.transform([[30,87000]])))

y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)

from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop
= X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000,
stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2,
classifier.predict(sc.transform(np.array([X1.ravel(),
X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('salmon',
'dodgerblue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c =
ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Random Forest Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
```

```
plt.legend()
plt.show()

from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop
= X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000,
stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2,
classifier.predict(sc.transform(np.array([X1.ravel(),
X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('salmon',
'dodgerblue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c =
ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Random Forest Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

4. KNN

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from math import sqrt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from matplotlib.colors import ListedColormap

dataset = pd.read_csv('Social_Network_Ads.csv')
dataset.info()

X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.25, random_state = 0)

print(X_train[:10])
print(y_train)
print(X_test[:10])
print(y_test)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test) #avoid data leakage

print(X_train[:10])
print(X_test.dtype)

class KNN():
  def __init__(self,k):
    self.k=k
    print(self.k)

  def fit(self,X_train,y_train):
    self.x_train=X_train
    self.y_train=y_train

  def calculate_euclidean(self,sample1,sample2):
    distance=0.0
    for i in range(len(sample1)):
      distance+=(sample1[i]-sample2[i])**2 #Euclidean Distance =
sqrt(sum i to N (x1_i - x2_i)^2)
    return sqrt(distance)

  def nearest_neighbors(self,test_sample):
    distances=[]#calculate distances from a test sample to every
sample in a training set
    for i in range(len(self.x_train)):

distances.append((self.y_train[i],self.calculate_euclidean(self.x_tr
ain[i],test_sample)))
```

```
    distances.sort(key=lambda x:x[1])#sort in ascending order, based
on a distance value
    neighbors=[]
    for i in range(self.k): #get first k samples
      neighbors.append(distances[i][0])
    return neighbors


  def predict(self,test_set):
    predictions=[]
    for test_sample in test_set:
      neighbors=self.nearest_neighbors(test_sample)
      labels=[sample for sample in neighbors]
      prediction=max(labels,key=labels.count)
      predictions.append(prediction)
    return predictions

model=KNN(5)
model.fit(X_train,y_train)

classifier = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2)
#The default metric is minkowski, and with p=2 is equivalent to the
standard Euclidean metric.
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
predictions=model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)

cm = confusion_matrix(y_test, predictions) #our model
print(cm)
accuracy_score(y_test, predictions)

X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop
= X_set[:, 0].max() + 10, step = 1),
```

```
                        np.arange(start = X_set[:, 1].min() - 1000,
    stop = X_set[:, 1].max() + 1000, step = 1))
    plt.contourf(X1, X2,
    classifier.predict(sc.transform(np.array([X1.ravel(),
    X2.ravel()]).T)).reshape(X1.shape),
                 alpha = 0.75, cmap = ListedColormap(('blue', 'green')))
    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())
    for i, j in enumerate(np.unique(y_set)):
        plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c =
    ListedColormap(('blue', 'green'))(i), label = j)
    plt.title('K-NN (Test set)')
    plt.xlabel('Age')
    plt.ylabel('Estimated Salary')
    plt.legend()
    plt.show()
```

Output:

1. **SVM**

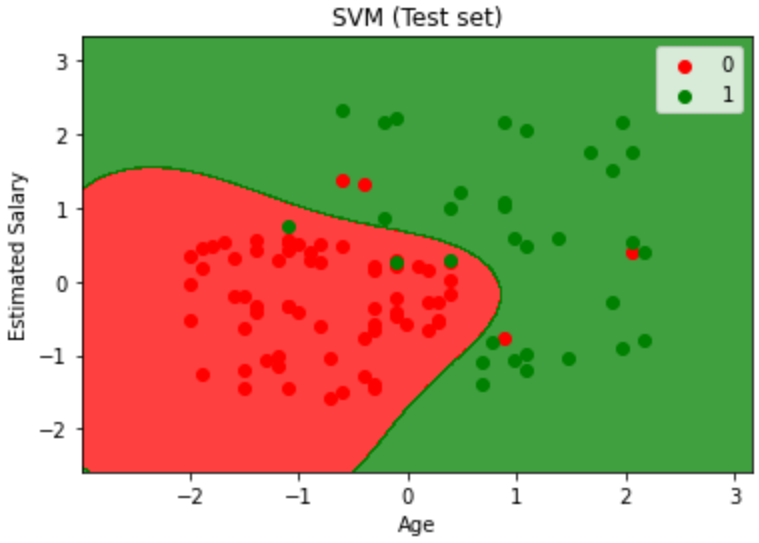
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

Confusion Matrix

```
[[64  4]
 [ 3 29]]
```

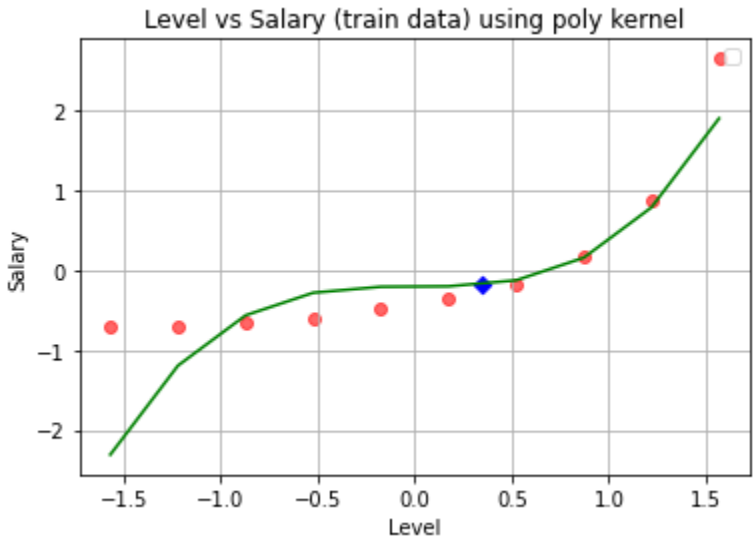
Accuracy

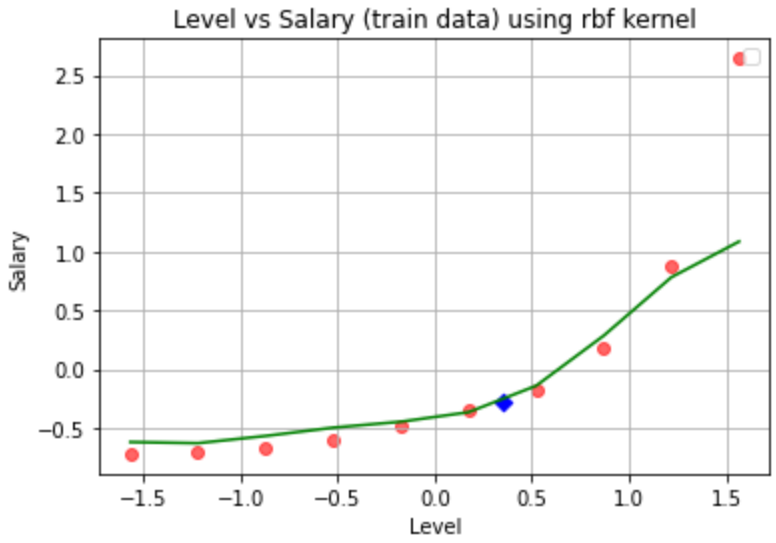
```
0.93
```



2. SVR

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Position  10 non-null     object
 1   Level     10 non-null     int64
 2   Salary    10 non-null     int64
dtypes: int64(2), object(1)
memory usage: 368.0+ bytes
```



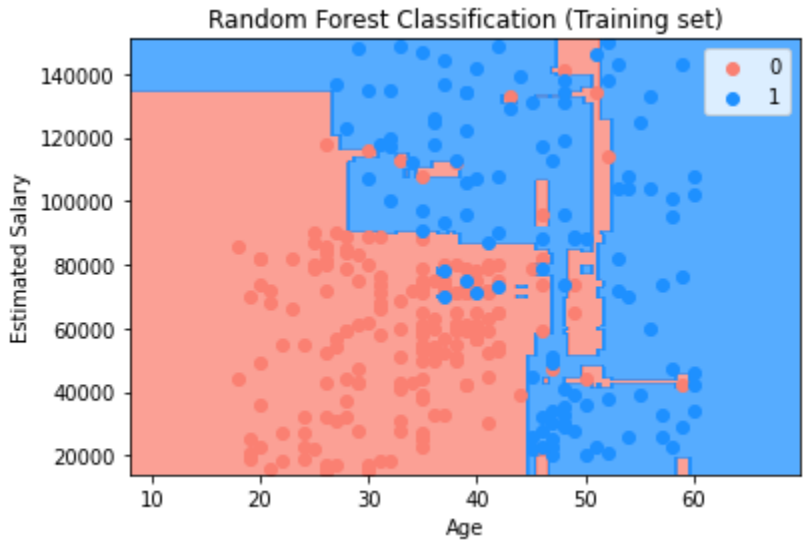


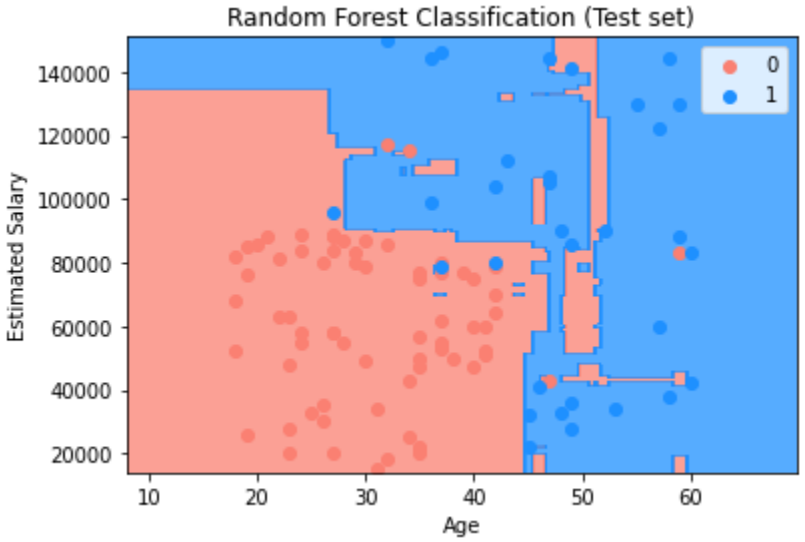
3. Random Forest

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
[[    44  39000]
 [    32 120000]
 [    38  61000]
 [    36  50000]
 [    36  63000]]

[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1
0 0 1 0 0 0 0]

[[    30  87000]
 [    38  50000]
 [    35  75000]
 [    47  43000]]

[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]

[[ 0.58164944 -0.88670699]
```

```
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]

RandomForestClassifier(criterion='entropy', n_estimators=10,
random_state=0)

[[63  5]
 [ 4 28]]
0.91
```

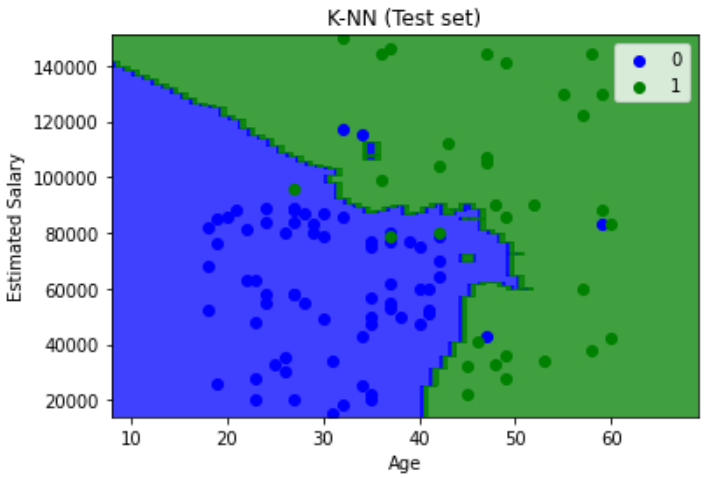




4. KNN

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

```
[[    44  39000]
 [    32 120000]
 [    38  61000]
 [    36  50000]
 [    36  63000]]

[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1
0 0 1 0 0 0 0]

[[    30  87000]
 [    38  50000]
 [    35  75000]
 [    47  43000]]

[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]

Float64

[[64  4]
 [ 3 29]]
0.93
```



Interpretation:

1. Advantages of Random Forest includes Random Forest is capable of performing both Classification and Regression tasks, It is capable of handling large datasets with high dimensionality, It enhances the accuracy of the model and prevents the overfitting issue.
2. Disadvantages of Random Forest include that although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.
3. Advantages of KNN Algorithm include It is simple to implement, It is robust to the noisy training data, It can be more effective if the training data is large.
4. Disadvantages of KNN Algorithm include that it always needs to determine the value of K which may be complex some time, the computation cost is high because of calculating the distance between the data points for all the training samples.
5. The main reason to use an SVM instead is because the problem might not be linearly separable. In that case, we will have to use an SVM with a non linear kernel.
6. Another related reason to use SVMs is if you are in a highly dimensional space. For example, SVMs have been reported to work better for text classification.
7. But it requires a lot of time for training. So, it is not recommended when we have a large number of training examples.
8. KNN is robust to noisy training data and is effective in case of large number of training examples.
9. But for this algorithm, we have to determine the value of parameter K (number of nearest neighbors) and the type of distance to be used. The computation time is also very much as we need to compute distance of each query instance to all training samples.
10. Random Forest is nothing more than a bunch of Decision Trees combined. They can handle categorical features very well.
11. This algorithm can handle high dimensional spaces as well as large number of training examples.
12. Random Forests can almost work out of the box and that is one reason why they are very popular.

Conclusion:

1. Thus we have implemented the different classifiers on different dataset and successfully performed the experiment.
2. If training data is much larger than no. of features, KNN is better than SVM. SVM outperforms KNN when there are large features and lesser training data. Both perform well when the training data is less, and there are large number of features.
3. Based on the various classifiers, learned the identifying features, advantages as well as the disadvantages of each of the implemented models.