Decision Tree : Cancer_Data.csv

K-Means : Mall_Customers.csv

KNN classification : Social_network_ads(1).CSV

LinearRegression : Salary_data.csv

Logistic Regression: Titanic.csv

Naive Bayes : Social_network_ads

Random_forest : Cancer_Data.csv

SVM : breast-cancer.csv

Naive Bayes.

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

In [2]:

```
dataset = pd.read_csv('Social_Network_Ads (1).csv')
x = dataset.iloc[:,[2, 3]].values
y = dataset.iloc[:, -1].values
```

In [3]:

```
print(dataset)
```

In [4]:

```
dataset.shape
```

In [5]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.30,random_state = 0)
```

In [6]:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

In [7]:

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

In [8]:

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

In [9]:

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop =
x_set[:, 0].max() + 1, step = 0.5),
                    np.arange(start = x_set[:, 1].min() - 1, stop =
x_set[:, 1].max() + 1, step = 0.5))

plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
             alpha = 0.75, cmap = ListedColormap(('white', 'black')))
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('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

In [10]:

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

In [11]:

```
y_pred
```

Out[11]:

In [12]:

```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test,
y_pred)))
```

```
Model accuracy score: 0.8917
```

SVM(Ideal)

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

In [45]:

```
df= pd.read_csv('breast-cancer.csv')
```

In [46]:

```
df.info()
```

In [47]:

```
import plotly.express as px
```

In [48]:

```
px.scatter(data_frame=df,x='symmetry_worst',color='diagnosis',color_discret
e_sequence=['#05445E','#75E6DA'])
```

In [49]:

```
df.drop('id', axis=1, inplace=True)
```

In [51]:

```
df.describe().T
```

In [52]:

```
df['diagnosis'] = (df['diagnosis'] == 'M').astype(int)
```

In [54]:

```
df
```

In [55]:

```
cor_target = abs(corr["diagnosis"])
relevant_features = cor_target[cor_target>0.2]
names = [index for index, value in relevant_features.iteritems()]
names.remove('diagnosis')
print(names)
```

In [56]:

```
X = df[names].values
y = df['diagnosis']
```

In [ ]:

In [62]:

```
def scale(X):

    mean = nm.mean(X, axis=0)
    std = nm.std(X, axis=0)

    X = (X - mean) / std
    return X
```

In [64]:

```
X = scale(X)
X
```

In [65]:

```
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_st
ate=2)
```

In [68]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

skmodel = SVC()
skmodel.fit(X_train, y_train)
sk_predictions = skmodel.predict(X_test)

print("Accuracy=", accuracy_score(y_test, sk_predictions))
```

```
Accuracy= 0.9790209790209791
```

In [69]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, sk_predictions)
print(cm)
```

In [73]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, sk_predictions))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98        87
           1       0.96      0.98      0.97        56

    accuracy                           0.98       143
   macro avg       0.98      0.98      0.98       143
weighted avg       0.98      0.98      0.98       143
```

Decision Tree

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

In [10]:

```
df= pd.read_csv('Cancer_Data.csv')
```

In [11]:

```
df.shape
```

In [12]:

```
df.head()
```

In [13]:

```
df.info()
```

In [14]:

```
import plotly.express as px
```

In [16]:

```
px.scatter(data_frame=df,x='symmetry_worst',color='diagnosis',color_discret
e_sequence=['#FFD54F','#1565C0'])
```

In [17]:

```
df.drop('id', axis=1, inplace=True)
```

In [18]:

```
df.describe().T
```

In [19]:

```
df['diagnosis'] = (df['diagnosis'] == 'M').astype(int)
```

In [20]:

```
df
```

In [22]:

```
corr = df.corr()
```

In [23]:

```
cor_target = abs(corr["diagnosis"])
relevant_features = cor_target[cor_target>0.2]
names = [index for index, value in relevant_features.iteritems()]
names.remove('diagnosis')
print(names)
```

In [24]:

```
X = df[names].values
y = df['diagnosis']
```

In [25]:

```
def scale(X):

    mean = nm.mean(X, axis=0)
    std = nm.std(X, axis=0)

    X = (X - mean) / std
    return X
```

In [26]:

```
X = scale(X)
X
```

In [27]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_sta
te=1)
```

In [28]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

skmodel = DecisionTreeClassifier()
skmodel.fit(X_train, y_train)
sk_predictions = skmodel.predict(X_test)

print("Accuracy=", accuracy_score(y_test, sk_predictions))
```

```
Accuracy= 0.956140350877193
```

In [29]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, sk_predictions)
print(cm)
```

In [30]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, sk_predictions))
```

K Means

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as py
import plotly.graph_objs as go
from sklearn.cluster import Kmeans

py.offline.init_notebook_mode(connected = True)
```

In [2]:

```
df = pd.read_csv('Mall_Customers.csv')
df.head()
```

In [3]:

```
df.describe()
```

In [4]:

```
df.dtypes
```

In [5]:

```
plt.figure(1 , figsize = (15 , 6))
for gender in ['Male' , 'Female']:
    plt.scatter(x = 'Age' , y = 'Annual Income (k$)' , data =
df[df['Gender'] == gender] ,
                s = 200 , alpha = 0.5 , label = gender)
plt.xlabel('Age'), plt.ylabel('Annual Income (k$)')
plt.title('Age vs Annual Income w.r.t Gender')
plt.legend()
plt.show()
```

In [6]:

```
plt.figure(1 , figsize = (15 , 6))
for gender in ['Male' , 'Female']:
    plt.scatter(x = 'Annual Income (k$)',y = 'Spending Score (1-100)' ,
                data = df[df['Gender'] == gender] ,s = 200 , alpha = 0.5 ,
label = gender)
plt.xlabel('Annual Income (k$)'), plt.ylabel('Spending Score (1-100)')
plt.title('Annual Income vs Spending Score w.r.t Gender')
plt.legend()
plt.show()
```

In [7]:

```
X1 = df[['Age' , 'Spending Score (1-100)']].iloc[: , :].values
inertia = []
for n in range(1 , 11):
    algorithm = (Kmeans(n_clusters = n ,init='k-means++', n_init = 10
,max_iter=300,
                        tol=0.0001,  random_state= 111  ,
algorithm='elkan') )
    algorithm.fit(X1)
    inertia.append(algorithm.inertia_)
```

In [8]:

```
algorithm = (Kmeans(n_clusters = 4 ,init='k-means++', n_init = 10
,max_iter=300,
                        tol=0.0001,  random_state= 111  ,
algorithm='elkan') )
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_
```

In [9]:

```
h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() + 1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))
Z = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [10]:

```
plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z = Z.reshape(xx.shape)
plt.imshow(Z , interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Age' ,y = 'Spending Score (1-100)' , data = df , c =
labels1 ,
            s = 200 )
plt.scatter(x = centroids1[: , 0] , y =  centroids1[: , 1] , s = 300 , c =
'red' , alpha = 0.5)
plt.ylabel('Spending Score (1-100)') , plt.xlabel('Age')
plt.show()
```

In [11]:

```
X2 = df[['Annual Income (k$)' , 'Spending Score (1-100)']].iloc[: ,
:].values
inertia = []
for n in range(1 , 11):
    algorithm = (Kmeans(n_clusters = n ,init='k-means++', n_init = 10
,max_iter=300,
                        tol=0.0001,  random_state= 111  ,
algorithm='elkan') )
    algorithm.fit(X2)
```

In [12]:

```
algorithm = (Kmeans(n_clusters = 5 ,init='k-means++', n_init = 10
,max_iter=300,
                        tol=0.0001,  random_state= 111  ,
algorithm='elkan') )
algorithm.fit(X2)
labels2 = algorithm.labels_
centroids2 = algorithm.cluster_centers_
```

In [13]:

```
h = 0.02
x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() + 1
y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))
Z2 = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [14]:

```
plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z2 = Z2.reshape(xx.shape)
plt.imshow(Z2 , interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Annual Income (k$)' ,y = 'Spending Score (1-100)' , data
= df , c = labels2 ,
            s = 200 )
plt.scatter(x = centroids2[: , 0] , y =  centroids2[: , 1] , s = 300 , c =
'red' , alpha = 0.5)
```

```
plt.ylabel('Spending Score (1-100)') , plt.xlabel('Annual Income (k$)')
plt.show()
```

In [15]:

```
X3 = df[['Age' , 'Annual Income (k$)' ,'Spending Score (1-100)']].iloc[: ,
:].values
inertia = []
for n in range(1 , 11):
    algorithm = (Kmeans(n_clusters = n ,init='k-means++', n_init = 10
,max_iter=300,
                        tol=0.0001,  random_state= 111  ,
algorithm='elkan') )
    algorithm.fit(X3)
    inertia.append(algorithm.inertia_)
```

In [16]:

```
algorithm = (Kmeans(n_clusters = 6 ,init='k-means++', n_init = 10
,max_iter=300,
                        tol=0.0001,  random_state= 111  ,
algorithm='elkan') )
algorithm.fit(X3)
labels3 = algorithm.labels_
centroids3 = algorithm.cluster_centers_
```

In [18]:

```
df['label3'] =  labels3
trace1 = go.Scatter3d(
    x= df['Age'],
    y= df['Spending Score (1-100)'],
    z= df['Annual Income (k$)'],
    mode='markers',
     marker=dict(
        color = df['label3'],
        size= 20,
        line=dict(
            color= df['label3'],
            width= 12
        ),
        opacity=0.8
     )
)
data = [trace1]
layout = go.Layout(
#     margin=dict(
#         l=0,
#         r=0,
#         b=0,
#         t=0
#     )
    title= 'Clusters',
    scene = dict(
            xaxis = dict(title  = 'Age'),
            yaxis = dict(title  = 'Spending Score'),
            zaxis = dict(title  = 'Annual Income')
        )
)
fig = go.Figure(data=data, layout=layout)
```

```
py.offline.iplot(fig)
```

KNN Classifier

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

In [29]:

```
df= pd.read_csv('Social_Network_Ads (1).csv')
```

In [30]:

```
x = df.iloc[:, [2, 3]].values
y = df.iloc[:, -1].values
```

In [31]:

```
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)
```

In [32]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [33]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)
```

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

In [35]:

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

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop =
x_set[:, 0].max() + 1, step = 0.01),
                     nm.arange(start = x_set[:, 1].min() - 1, stop =
x_set[:, 1].max() + 1, step = 0.01))
plt.contourf(x1, x2, knn.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
             alpha = 0.75, cmap = ListedColormap(('white', 'black')))
plt.xlim(x1.min(), x1.max())
plt.xlim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
```

```
                c = ListedColormap(('black', 'white'))(i), label = j)
plt.title('KNN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Linear Regression

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

In [2]:

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

In [3]:

```
dataset.head()
```


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

In [2]:

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

In [3]:

```
dataset.head()
```


```
y_pred = regressor.predict(X_test)
```

In [7]:

```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Logistic Regression

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

In [2]:

```
dataset = pd.read_csv('titanic.csv')
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

In [3]:

```
dataset.tail()
```

```
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)
```

In [5]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [6]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

In [7]:

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

In [8]:

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

In [9]:

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
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(('white', 'black')))
plt.xlim(x1.min(), x1.max())
plt.xlim(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(('black', 'white'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Random Forest

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

In [2]:

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

In [3]:

```
df.info()
```

In [4]:

```
import plotly.express as px
```

In [5]:

```
px.scatter(data_frame=df,x='symmetry_worst',color='diagnosis',color_discret
e_sequence=['#AA00FF','#00E676'])
```

In [6]:

```
df.drop('id', axis=1, inplace=True)
```

In [7]:

```
df.describe().T
```

```
df['diagnosis'] = (df['diagnosis'] == 'M').astype(int)
```

In [9]:

```
corr = df.corr()
```

In [10]:

```
cor_target = abs(corr["diagnosis"])
relevant_features = cor_target[cor_target>0.2]
names = [index for index, value in relevant_features.iteritems()]
names.remove('diagnosis')
print(names)
```

```
X = df[names].values
y = df['diagnosis']
```

In [12]:

```
def scale(X):

    mean = nm.mean(X, axis=0)
    std = nm.std(X, axis=0)

    X = (X - mean) / std
    return X
```

In [13]:

```
X = scale(X)
X
```

In [14]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_st
ate=3)
```

In [15]:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

skmodel = RandomForestClassifier()
skmodel.fit(X_train, y_train)
sk_predictions = skmodel.predict(X_test)

print("Accuracy=", accuracy_score(y_test, sk_predictions))
```

```
Accuracy= 0.9385964912280702
```

In [16]:

```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, sk_predictions)
print(cm)
```

In [17]:

```python
from sklearn.metrics import classification_report
print(classification_report(y_test, sk_predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95        74
           1       0.92      0.90      0.91        40

    accuracy                           0.94       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114
```