Knearest-neighbours

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
ds= pd.read_csv('Social_Network_Ads.csv')
x=ds.iloc[:,:-1].values
y=ds.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)

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

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, metric="minkowski", p=2)
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)
accuracy_score(y_test,y_pred)

from matplotlib.colors import ListedColormap
x_set, y_set= sc.inverser_tranform(x_train), y_train
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(('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('knn training set')
plt.xlabel('age')
plt.ylabel('estimated salary')
plt.legend()
plt.show()

from matplotlib.colors import ListedColormap
x_set, y_set= sc.inverser_tranform(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(('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('knn test 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

dataset = pd.read_csv('Salary_Data.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 = 1/3, random_state = 0)

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

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

Multiple Linear Regression

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

dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))

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_state = 0)

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

KMeans

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
ds = pd.read_csv('Mall_Customers.csv')
x=ds.iloc[:,[3,4]].values

from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
      kmeans=Kmeans(n_clusters=i, init='k-means++', random_state=42)
      kmeans.fit(x)
      wcss.append(kmeans.inertia_)
plt.plot(range(1,11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

kmeans=Kmeans(n_clusters=5, init= 'k-means++', random_state= 42)
y_kmeans= kmeans.fit_predict(x)

plt.scatter(x[y_kmeans==0,0], x[y_kmeans == 0,1], s=100, c='red' label='cluster 1')
plt.scatter(x[y_kmeans==1,0], x[y_kmeans== 1,1], s=100, c='blue', label='cluster 2')
plt.scatter(x[y_kmeans==2,0], x[y_kmeans == 2,1], s=100, c='green' label='cluster 3')
plt.scatter(x[y_kmeans==3,0], x[y_kmeans== 3,1], s=100, c='yellow', label='cluster 4')
plt.scatter(x[y_kmeans==4,0], x[y_kmeans== 4,1], s=100, c='cyan', label='cluster 4')

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=300, c= 'yellow',
label='Centroids')
plt.title('Clusters of customers')
plt.xlabel("Annual income k$")
plt.ylabel('Spending score(1-100)')
plt.legend()
plt.show()
```

Heirarchical Clustering

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
ds= pd.read_csv('Mall_Customers.csv')
x=ds.iloc[:,[3,4]].values

import scipy.cluster.hierarchy as sch
dendogram = sch.dendogram(sch.linkage(x, method='ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Eucidean distances')
plt.show()

from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeCLustering(n_clusters=5, affinity='euclidean', linkage='ward')
y_hc= hc.fit_predict(x)

plt.scatter(x[y_hc== 0,0], x[y_hc==0,1], s=100, c='red', label='Cluster 1')
plt.scatter(x[y_hc== 1,0], x[y_hc==1,1], s=100, c='blue', label='cluster 2')
plt.scatter(x[y_hc== 2,0], x[y_hc==2,1], s=100, c='green', label='Cluster 3')
plt.scatter(x[y_hc== 3,0], x[y_hc==3,1], s=100, c='cyan', label='cluster 4')
plt.scatter(x[y_hc== 4,0], x[y_hc==4,1], s=100, c='magenta', label='Cluster 5')

plt.title('Clusters of customers')
plt.xlabel('Annual Income(k$)')
plt.ylabel('Spending score(1-100)')
plt.legend()
plt.show()
```

ANN

```
import numpy as np
import pandas as pd
import tensorflow as tf

ds= pd.read_csv('Churn_Modelling.csv')
x=ds.iloc[:,3:-1].values
y=ds.iloc[:,-1].values

from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
x[:,2]=le.fit_transform(x[:,2])


from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct= ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])],
remainder="passthrough")
x=np.array(ct.fit_transform(x))

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_state=0)

ann= tf.keras.models.Sequential()
ann.add(tf.keras.layers.Denser(units=6, activation='relu'))

ann.add(tf.keras.layers.Denser(units=6, activation = 'relu'))
ann.add(tf.keras.layers.Denser(units=1, activation='sigmoid'))

ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
ann.fit(x_train, y_train, batch_size=32, epochs=100)

print(ann.predict(sc.transform([[1,0,0,0,600,1,40,3,60000,2,1,1,50000]]))> 0.5)

y_pred=ann.predict(x_test)
y_pred=(y_pred>0.5)
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)
```

SVM

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

ds= pd.read_csv('Social_Network_Ads.csv')
x=ds.iloc[:,:-1].values
y=ds.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=0.5, random_state=0)

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

from sklearn.svm import svc
classifier = svc(kernel='linear', random_state=0)
classifier.fit(x_train, y_train)
print(classifer.predict(sc.transform([[30,87000]])))
y_pred= classifer.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)
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.arrange(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, classifer.predict(sc.transform(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(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.arrange(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, classifer.predict(sc.transform(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()
```