Simple Linear Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('F:/ML\_Programs/Salary\_Data.csv')

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

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

# Splitting the dataset into the Training set and Test set

fromsklearn.model\_selectionimporttrain\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0)

# Training the Simple Linear Regression model on the Training set

fromsklearn.linear\_modelimportLinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = regressor.predict(X\_test)

# Visualising the Training set results

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()

# Visualising the Test set results

plt.scatter(X\_test, y\_test, color = 'red')

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Test set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

Multiple Linear Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('F:/ML\_Programs/50\_Startups.csv')

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

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

#print(X)

# Encoding categorical data

fromsklearn.composeimportColumnTransformer

fromsklearn.preprocessingimportOneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

# Splitting the dataset into the Training set and Test set

fromsklearn.model\_selectionimporttrain\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# Training the Multiple Linear Regression model on the Training set

fromsklearn.linear\_modelimportLinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

# Predicting the Test set results

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))

Logistic Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('F:/ML\_Programs/Social\_Network\_Ads.csv')

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

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

# Splitting the dataset into the Training set and Test set

fromsklearn.model\_selectionimporttrain\_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)

# Feature Scaling

fromsklearn.preprocessingimportStandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

# Training the Logistic Regression model on the Training set

fromsklearn.linear\_modelimportLogisticRegression

classifier = LogisticRegression(random\_state=0)

classifier.fit(X\_train, y\_train)

#Predict the result for Age = 30 and EstimatedSalary = 87000

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

#Predicting the Test Set results

y\_pred = classifier.predict(X\_test)

print(y\_pred)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1),

                      y\_test.reshape(len(y\_test),1)),1))

# Making the Confusion Matrix

fromsklearn.metricsimportconfusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

# Visualising the Training set results

frommatplotlib.colorsimportListedColormap

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(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

fori, 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('Logistic Regression (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# Visualising the Test set results

frommatplotlib.colorsimportListedColormap

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(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

fori, 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('Logistic Regression (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

## 

## Neural Network Program

fromgoogle.colabimport drive

drive.mount('/content/drive')

# first neural network with keras tutorial

from numpy importloadtxt

fromkeras.modelsimport Sequential

fromkeras.layersimport Dense

# load the dataset

dataset = loadtxt('/content/drive/My Drive/pima-indians-diabetes.csv', delimiter=',')

# split into input (X) and output (y) variables

X = dataset[:,0:8]

y = dataset[:,8]

dataset.shape

X.shape

y.shape

# define the keras model

model = Sequential() #calling default constructor

#First Hidden Layer (along with input layer)

model.add(Dense(12, input\_dim=8, activation='relu'))  #adding input layer along with the 1st hidden layer

#1st hidden layer has 12 activation units, input layer has 8 units for 8 features

#activation function used in the 1st layer is ReLU

#Adding the 2nd Hidden layer with 8 activation nodes and ReLU function

model.add(Dense(12, activation='relu'))

#Adding output layer with 1 activation unit (binary classification) and with Sigmoid function

model.add(Dense(1, activation='sigmoid'))

# compile the keras model

#Loss means computing the error or cost function (using algorithm - binary\_crossentropy)

#Algorithm used is 'adam' - Stochastic gradient descent algorithm

#Evaluating the performance of the model will be done using 'accuracy'

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the keras model on the dataset

# Batch size is for updating the weights. Weights are updated after every 10 rows are executed.

model.fit(X, y, epochs=200, batch\_size=10)

# evaluate the keras model

\_, accuracy = model.evaluate(X, y)

print('Accuracy: %.2f' % (accuracy\*100))

# make probability predictions with the model

predictions = model.predict(X)

# round predictions

rounded = [round(x[0]) for x in predictions]

# make class predictions with the model

predictions = model.predict\_classes(X)

predictions

# make class predictions with the model

predictions = model.predict\_classes(X)

# summarize the first 5 cases

foriin range(5):

    print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))

Support Vector Machine (SVM)

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('F:/MachineLearning/Social\_Network\_Ads.csv')

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

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

# Splitting the dataset into the Training set and Test set

fromsklearn.model\_selectionimporttrain\_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)

# Feature Scaling

fromsklearn.preprocessingimportStandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

# Training the SVM model on the Training set

fromsklearn.svmimport SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting a new result

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

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

# Making the Confusion Matrix

fromsklearn.metricsimportconfusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

# Visualising the Training set results

frommatplotlib.colorsimportListedColormap

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(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

fori, 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()

# Visualising the Test set results

frommatplotlib.colorsimportListedColormap

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(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

fori, 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()

Convolutional Neural Network

# Importing the libraries

importtensorflow as tf

fromkeras.preprocessing.image import ImageDataGenerator

tf.\_\_version\_\_

Part 1 - Data Preprocessing

# Preprocessing the Training set

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

training\_set = train\_datagen.flow\_from\_directory('/content/drive/MyDrive/cnn/training\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

# Preprocessing the Test set

test\_datagen = ImageDataGenerator(rescale = 1./255)

test\_set = test\_datagen.flow\_from\_directory('/content/drive/MyDrive/cnn/testing\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

fromgoogle.colab import drive

drive.mount('/content/drive')

Part 2 - Building the CNN

# Initialising the CNN

cnn = tf.keras.models.Sequential()

# Step 1 - Convolution

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu', input\_shape=[64, 64, 3]))

# Step 2 - Pooling

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

# Adding a second convolutional layer

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

# Step 3 - Flattening

cnn.add(tf.keras.layers.Flatten())

# Step 4 - Full Connection

cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

# Step 5 - Output Layer

cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

Part 3 - Training the CNN

# Compiling the CNN

cnn.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Training the CNN on the Training set and evaluating it on the Test set

cnn.fit(x = training\_set, validation\_data = test\_set, epochs = 10)

Part 4 - Making a single prediction

import numpy as np

fromkeras.preprocessing import image

test\_image = image.load\_img('/content/drive/MyDrive/cnn/200.jpg', target\_size = (64, 64))

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = cnn.predict(test\_image)

training\_set.class\_indices

if result[0][0] == 1:

prediction = '2000 note'

else:

prediction = '200 note'

print(prediction)