# Super Data Science Notes

Some of this is not from SDS but included from my other classes.

## Linear Regression

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

dataset = pd.read\_excel('data.xlsx')

dataset.head()

dataset.describe()

plt.figure(dpi=150)

sns.pairplot(data=dataset)

**Separate X (independent) and y (dependent) variables**

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

# y = dataset['PE']

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

**Separate Train and Test Sets**

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)

X\_train

X\_test

y\_train

y\_test

**Build and Train Model**

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

## SDS Linear Regression

## Part 1 Data Preprocessing

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import sweetviz

### Importing the Dataset

dataset = pd.read\_excel('data.xlsx')

dataset.head()

dataset.describe()

len(dataset)

plt.figure(dpi=150)

sns.pairplot(data=dataset)

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

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

report = sweetviz.analyze(dataset, target\_feat="PE")

# report.show\_html(layout="vertical")

# report.show\_notebook()

## Creating the Training and Test Sets

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=11)

X\_train

X\_test

y\_train

y\_test

## Build and Train Model

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

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

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

y\_pred

## Single Record Prediction

# AT = 15, V = 40, AP = 1000, RH = 75 - totally new data

model.predict([[15, 40, 1000, 75]])

## Model Evaluation

### R Squared

from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

r2

### Adjusted R Squared

k = X\_test.shape[1]

print(k)

n = X\_test.shape[0]

print(n)

adj\_r2 = 1-(1-r2)\*(n-1)/(n-k-1)

print(adj\_r2)

### Other Model Evaluations

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

mean\_absolute\_error(y\_test, y\_pred)

mean\_squared\_error(y\_test, y\_pred)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

## SDS Logistic Regression – Classification

Classification - Logistic Regression

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import sweetviz

# Part 1 - Data Preprocessing

# Import the Dataset

dataset = pd.read\_csv('data.csv')

dataset.head(10)

dataset.describe()

dataset.info()

# Train Test Split

X = dataset.iloc[:, 1:-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=0.2, random\_state=0)

X\_train

X\_test

y\_train

y\_test

# Feature Scaling

# Normalization values convert to between 0 and 1

# Standardization values convert to relevant standrd deviation outliers more than 3

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

# Part 2 - Building and Training the Model

# Building the Model

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression(random\_state = 0)

# Traing the Model

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

# Inference

y\_pred = model.predict(sc.transform(X\_test))

y\_pred

# Predict a single value against the model

model.predict(sc.transform([[1,2,3,4,5,6,7,8,9]]))

# Part 3 - Evaluating the Model

# Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score, ConfusionMatrixDisplay, plot\_confusion\_matrix

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

# Accuracy

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

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

plot\_confusion\_matrix(model, scaled\_X\_test, y\_test )

## Unsupervised Learning

## K Means Clustering

# Import the Dataset

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import sweetviz

dataset = pd.read\_csv('km\_data.csv')

dataset.head()

len(dataset)

report = sweetviz.analyze(dataset, target\_feat="Spending Score (1-100)")

report.show\_html(layout="vertical")

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

X

# Use elbow method to find optimal number of clusters

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()

# Train the K-Means Model

kmeans = KMeans(n\_clusters=5, init='k-means++', random\_state=42)

y\_pred = kmeans.fit\_predict(X)

y\_pred

# Can I join this to my dataset

X

# Visual the Clusters

plt.scatter(X[y\_pred == 0, 0], X[y\_pred == 0, 1], s = 100, c='red', label = 'Cluster 0')

plt.scatter(X[y\_pred == 1, 0], X[y\_pred == 1, 1], s = 100, c='blue', label = 'Cluster 1')

plt.scatter(X[y\_pred == 2, 0], X[y\_pred == 2, 1], s = 100, c='green', label = 'Cluster 2')

plt.scatter(X[y\_pred == 3, 0], X[y\_pred == 3, 1], s = 100, c='cyan', label = 'Cluster 3')

plt.scatter(X[y\_pred == 4, 0], X[y\_pred == 4, 1], s = 100, c='magenta', label = 'Cluster 4')

plt.scatter(kmeans.cluster\_centers\_[:,0], kmeans.cluster\_centers\_[:,1], s = 300, c='yellow', label = 'Centroids')

plt.title('Clusteres of Customers')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.legend()

plt.show()

# My Try

# Predict a single new value

my\_pred = kmeans.predict([[16, 6]])

my\_pred

# dataset = pd.DataFrame(np.hstack((images, label.reshape(-1, 1))))

my\_dataset = pd.DataFrame(np.hstack((X, y\_pred.reshape(-1, 1))))

my\_dataset.head()