INDEX

NAME: Surya Bhat STD.: 6 SEC.: D ROLL NO.: 225 SUB.: ~~Software~~ ML Lab

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	21/3/24	Pandas (Reading csv)		4/4/24
2	28/3/24	Housing Prediction		18/4/24
3	18/4/24	Decision Tree		25/4
4	25/4/24	Linear & Logistic Regression		25/4
5	9/5/24	KNN		9/5/24
6	23/5/24	AdaBoost & Random Forest		
7	23/5/24	ANN		30/5/24
8	30/5/24	K-Means		30/5/24
9	30/5/24	PCA		

21/3/24

Week - 1

Pandas (Reading csv)

```
import pandas as pd

df2 = pd.read_csv("/content/sample_data/california_housing_test.csv")
df2.head()
```

Output:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
0	-122.05	37.37	27.0	3885.0	661.0

population	households	median_income	median_house_value
1537.0	606.0	6.6085	344700.0

```
df2.to_csv("hi.csv")

col_names = ['sepal_length_in_cm', 'sepal_width_in_cm', 'petal_length_in_cm',
             'petal_width_in_cm', 'class']

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/
       iris.data"
df = pd.read_csv(url, names = col_names)
df.head()
```

Output:

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm
0	5.1	3.5	1.4

petal_width_in_cm	class
0.2	Iris-setosa

21/3/24

28/3/24

Week-2

1. Performance Measure
2. Get the data

2.1. Download the data

```
import os
import tarfile
import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/
handson-ml2/master/"
HOUSING_PATH = os.path.join("data", "01")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url= HOUSING_URL, housing_path=
    HOUSING_PATH):

    os.makedirs(name = housing_path, exist_ok= True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(url = housing_url, filename = tgz_path)
    housing_tgz = tarfile.open(name = tgz_path)
    housing_tgz.extractall(path = housing_path)
    housing_tgz.close()
```

2.2 Take a quick look at the Data Structure

```
housing = load_housing_data()
housing.head()
housing.info()
housing.describe()
```

2.3 Plotting graphs

```
import matplotlib.pyplot as plt
import seaborn as sns

housing.hist(bins=50, figsize=(20,15))
plt.show()
```

2.4 Create test set

```
import numpy as np

def split_train_test(data, test_ratio=0.2)
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]

train_set, test_set = split_train_test(data = housing)
len(train_set), len(test_set)

from zlib import crc32
def test_set_check(identifier, test_ratio=0.2):
    total_size = 2**32
    hex_repr = crc32(np.int64(identifier)) & 0xffffffff
    in_test = hex_repr < (test_ratio * total_size)
    return in_test

[test_set_check(i) for i in range(10)]
```

2.5 Co-ordinates to Index

```
def from_Z_to_N(z):
    if z >= 0:
        n = 2*z
    else:
        n = -2*z - 1
    return n

def cantor_pairing(n1, n2):
    n = (((n1+n2)*(n1+n2+1))/2) + n2
    return n

def lat_lon_to_index(lat, lon):
    lat, lon = int(lat*100), int(lon*100)
    lat, lon = from_Z_to_N(lat), from_Z_to_N(lon)
    index = cantor_pairing(lat, lon)
    return np.int64(index)

housing['id'] = housing.apply(lambda row: lat_lon_to_index
  (row['latitude'], row['longitude']), axis=1)

housing['id'].value_counts()
```

28/3/24

28/3/24

Week - 2

3. Discover & visualise the data to gain insights

3.1

```
start_test_set.reset_index().to_flattner(frame='data/oi/start_test_set)

housing = start_train_set.copy
housing.shape
```

3.2 Visualising Geographical Data

```
housing.plot (kind='scatter', x='longitude', y='latitude')
plt.show()
```

3.3 Looking for correlations

```
corr_matrix = housing.corr()
corr_matrix['median_house_value'].sort_values(ascending=False)
```

3.4 Experimenting with attribute combinations

```
housing['rooms_per_household'] = housing['total_rooms']/housing['households']
housing['bedrooms_per_room'] = housing['total_bedrooms']/housing['total_rooms']
```

4. Prepare the data for machine learning

4.1 Data Cleaning

```
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fill(housing_num)
```

4.2 Handling Text and Categorical Attributes

```
housing_cat = housing['ocean_proximity']
housing_cat.head(10)
housing_cat['ocean_proximity'].value_counts()
```

4.3 Custom Transforms

```
class combinedAttributesAdder(BaseEstimator, TransformerMaxim):
    def __init__(self, add_bedrooms_per_room = True):
        self.add_bedrooms_per_room = add_bedrooms_per_room

    def fit(self, x, y=None):
        return self

    def transformer(self, x, y=None):
        rooms_per_household = X[:, rooms_ix]/x[:, households_ix]
        population_per_household = X[:, population_ix]/x[:, households_ix]
        return np.c_[x, rooms_per_household, population_per_
                household]
```

4.4 Transformation Pipelines

```
num_pipeline = Pipeline([('imputer', SimpleImputer(strategy='median')),
                         ('attribs_adder', combinedAttributesAdder()),
                         ('std_scaler', StandardScaler())
```

5. Train Model

```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Std:", scores.std())
```

6. Fine Tune Model

```
param_grid = [
    {'n_estimators': [3,10,30], 'max_features': [2,4,6,8]},
    {'bootstrap': [False], 'n_estimators': [3,10],
     max_features: [2,3,4]}
```

7. Evaluate on test set

```
final_model = grid_search.best_estimator

final_predictions = final_model.predict(X = X_test_prepared)

final_mse = mean_squared_error(y_true = y_test, y_pred =
                                final_predictions)

final_rmse = np.sqrt(final_mse)
final_rmse
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import random_state
from sklearn.linear_model import LinearRegression

df_sal = pd.read_csv("SalaryData.csv")
df_sal.head()

x = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,
        random_state=0)

model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
y_pred_train = model.predict(x_train)
plt.scatter(x_train, y_train, colour='lightcoral')
plt.scatter(x_train, y_pred_train, color='firebrick')
plt.scatter(x_test, y_test, color='lightcoral')
print(f"Coefficient = {model.coef()}")
print(f"Intercept = {model.intercept()}")
```

Output:

```
Coefficient = [[9312.57512673]]
Intercept = [26780.09915063]]
```

25/4/24

18/4/24

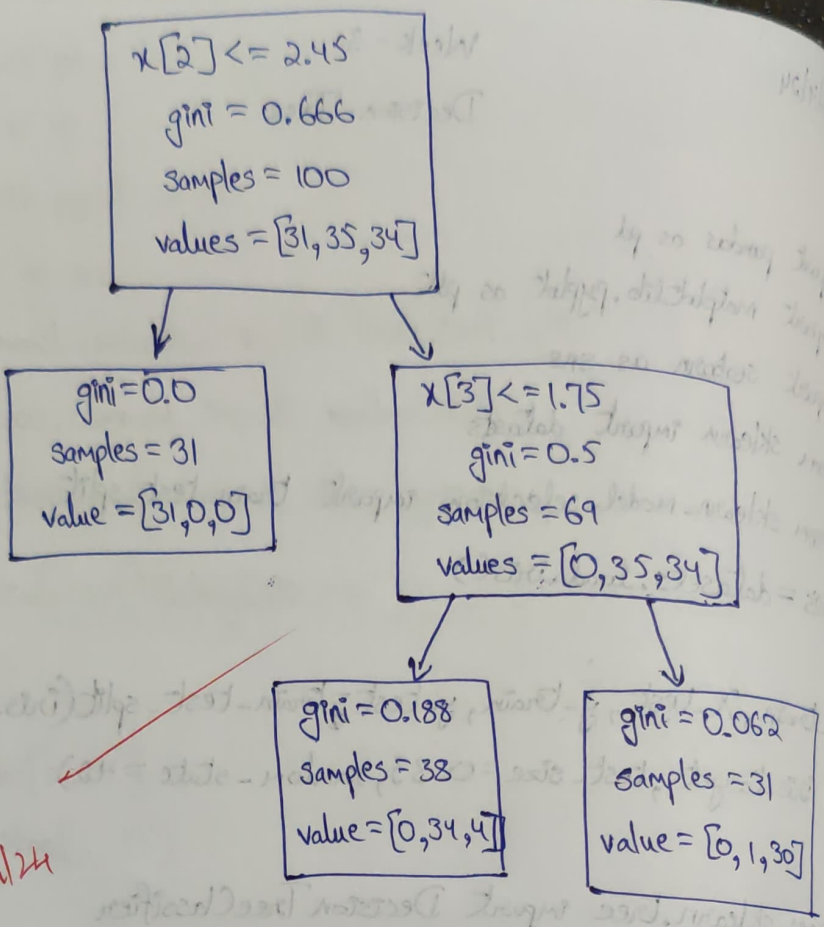
Week - 3

Decision Tree

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
iris = datasets.load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data,
    iris.target, test_size = 0.33, random_state = 42)

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(x_train, y_train)

from sklearn import tree
tree.plot_tree(model)

from sklearn.metrics import accuracy_score, classification_report
y_pred = model.predict(x_test)
print(classification_report(y_test, y_pred, target_names = ['0', '1', '2']))
```

Output:

0.98


x[2] <= 2.45
gini = 0.666
samples = 100
values = [31, 35, 34]
gini = 0.0
samples = 31
value = [31, 0, 0]
x[3] <= 1.75
gini = 0.5
samples = 69
values = [0, 35, 34]
gini = 0.188
samples = 38
value = [0, 34, 4]
gini = 0.062
samples = 31
value = [0, 1, 30]

25/11/24

25/4/24

Week - 4

Logistic Regression

```
import pandas as pd
from matplotlib.pyplot import pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression

df = pd.read_csv("insurance_data.csv")
df.head()

plt.scatter(df.age, df.bought_insurance, marker='+', color='red')

X_train, X_test, y_train, y_test = train_test_split(df[['age']],
    df.bought_insurance, train_size=0.7, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
model.predict_proba(X_test)
model.predict(X_test)
model.score(X_test, y_test)

model = LinearRegression()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

```
import math
def sigmoid(z):
    return 1/(1+math.exp(-z))

def predi(age):
    z = 0.042*age - 1.53
    y = sigmoid(z)
    return y

print(predi(35))
print(predi(43))
```

Output :

Prediction : array([1, 0, 1, 0, 0, 0, 0, 1, 0])

Score : 0.8888

Linear Reg Score : 0.584321

Predictions : 0.485

0.5685

25/4/24

9/5/24 Week-6

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, classification_report
from sklearn.neighbors import KNeighborsClassifier

col_names =["sepal_length_in_cm","sepal_width_in_cm","petal_length_in_cm",
"petal_width_in_cm","class"]
dataset = pd.read_csv("https://archive.ics.uci.edu/iris.data",header=
None, names = colnames)

dataset = dataset.replace({"class":{"Iris-setosa":1,"Iris-versicolor":2,
"Iris-virginica":3}})

dataset.head()

sns.heatmap(dataset.corr())
plt.title('Correlation On Iris Classes')

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

```
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(y_pred)
print(classification_report(y_test, y_pred, target_names = ['1','2','3']))

classifier2 = KNeighborsClassifier(n_neighbors = 2)
classifier2.fit(X_train, y_train)
y_pred = classifier2.predict(X_test)
print(y_pred)
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test, y_pred, target_names = ['1','2','3']))
```

Accuracy:

SVM : 0.98

KNN : 0.98

9/5/24

23/5/24 Week-7

```
import numpy as np
import pandas as pd

df = pd.read_csv("allergens.csv")
df.isnull().sum()
df['Prediction'].fillna(df.Prediction.mode()[0], inplace = True)
df.isnull().sum()
df.fillna(value='None', inplace = True)
df = df.drop('Allergen's', axis = 1)
y = df['Prediction']
x = df.drop('Prediction', axis = 1)
x = x.drop('Food Product', axis = 1)
x = x.apply(lambda y: ''.join(map(str,y)), axis = 1)

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, shuffle
= True, random_state = 42)

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
clf = Pipeline([
        ('vectorizer_tfidf', TfidfVectorizer()),
        ('AdaBoost', AdaBoostClassifier(n_estimators = 750, learning_rate
=0.8))
    ])

clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
```

```
print("Accuracy is ", accuracy_score(y_test, y_pred))

from sklearn.ensemble import RandomForestClassifier
clf2 = Pipeline([
    ('vectorizer_tfidf', TfidfVectorizer()),
    ('RF', RandomForestClassifier(n_estimators = 100, criterion='entropy'))
])
clf2.fit(x_train, y_train)
y_pred = clf2.predict(x_test)
print("Accuracy is ", accuracy_score(y_test, y_pred))
```

Output :

AdaBoost : 0.95

Random Forest : 0.9375

23/5/24

ANN

```
import numpy as np
X = np.array([[2,9],[1,5],[3,6]], dtype = float)
y = np.array([[92],[86],[89]], dtype = float)
X = X/np.amax(X, axis=0)
y = y/100
epoch = 500
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1
wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1,hiddenlayer_neurons))
wout = np.random_uniform(size=(hiddenlayer_neurons, output_neurons))
bout = np.random.uniform(size=(1, output_neurons))

def sigmoid(x):
    return 1/(1+np.exp(-x))

def derivatives_sigmoid(x):
    return x*(1-x)

for i in range(epoch):
    hinpl = np.dot(X, wh)
    hinp = hinpl + bh
    hlayer_act = sigmoid(hinp)
    outinpl = np.dot(hlayer_act, wout)
    outinp = outinpl + bout
    output = sigmoid(outinp)
```

```
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)

hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad

wout+= hlayer_act.T.dot(d_output) * lr
wh+= X.T.dot(d_hiddenlayer) * lr

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n", output)
```

```
Input:
[[0.6667  1.    ]
 [0.3333  0.5556]
 [1.      0.6667]]

Actual Output:
[[0.92]
 [0.86]
 [0.89]]

Predicted Output:
[[0.8509963 ]
 [0.83531048]
 [0.85823955]]
```

30/5/24

30/5/24

K-Means

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

model = KMeans(n_clusters = 3)
model.fit(X)

plt.figure(figsize =(14,14))
colormap = np.array(['red','lime','black'])
plt.subplot(2,2,1)
plt.scatter(X.Petal_Length, X.Petal_Width, c = colormap[y.Targets],s =40)
plt.title('Real Clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

plt.subplot(2,2,2)
plt.scatter(X.Petal_Length, X.Petal_Width, c = colormap[model.labels_],s =40)
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

30/5/24

PCA

```
import (
    matplotlib.pyplot as plt,
    pandas as pd,
    numpy as np,
    seaborn as sns)
%matplotlib inline
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(cancer['DESCR'])

from sklearn.preprocessing import StandardScaler,
scaler = StandardScaler()
scaler.fit(df)
scaled_data = scaler.transform(df)
~~from sklearn.prepo~~ from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)
scaled_data.shape
x_pca.shape
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0], x_pca[:,1], c=cancer['target'], cmap='plasma')
plt.xlabel('First Principle Component')
plt.ylabel('Second Principle Component')
```

Thurs 30/5/24