INDEX

Name SARAAG Sub. ML lab

Std.: 6C Div. Roll No. 190

Telephone No. E-mail ID.

Blood Group. Birth Day.

Sr.No.	Title	Date	Sign./Remarks
01.	Week-1' Import/Export using pandas library	21/3/24	
02.	W2: End-to-End ML	28/3/24	
03.	W3: Describe & Visualize Data	4/4/24	
04.	W4: Linear & Multiple Regression	15/4/24	
05.	W5: ID3 Decision Tree	25/4/24	
06.	W6: Logistic Regression & KNN Classification	9/5/24	
07.	W7: SVM, KMeans & PCA	23/5/24	
08.	W8: Random Forest & Boosting Ensemble Methods	30/5/24	

30/5/2024

Week - 1

21st March, 2024.

1) Python program to import and export data using pandas library.

```
import pandas as pd
airb_data = pd.read_csv("data/listing_austin.csv")
airb_data.head()
```

O/P:-

id	name	host_id	host_name	neighbourhood_grop	neighbour
2065	NW Austin	2466	paddy	NULL	787022

latitude	longitude	room_type	price
30.27752	-97.7137	private	179

2) Reading data from the URL

```
url = "https://archive.ics.uci.edu/ml/machine-learning/iris.data"
col_names = ["sepal_len_in_cm",
             "sepal_width_in_cm",
             "petal_len_in_cm",
             "petal_width_in_cm",
             "class"]

#Read data from URL
iris_data = pd.read_csv(url, names = col_names)

iris_data.head()
```

O/p:

	sepal_len_in_cm	sepal_width_in_cm	petal_len_in_cm	petal_width_in_cm class
0	5.1	3.5	1.4	0.2
1	4.9	3	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5	3.5	1.9	0.2

class

iris_setosa

iris_setosa

iris_setosa

iris_setosa

iris_setosa

3) Exporting data frame to a csv file

iris_data.to_csv("cleaned_iris_data.csv")

Week-2 23th March, 2024.

END TO END ML

* Get the data

① import os
import tarfile
import urllib

② Download_root = "https://raw.githubusercontent.com/ageron/ml2/master"
Housing_path = os.path.join("data", "01")
Housing_url = Download_root + "datasets/housing/housing.tgz"

③ 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.urlretrive(url=housing_url, filename=tgz_path)
housing_tgz = tarfile.open(name=tgz_path)
housing_tgz.extractall(path=housing_path)
housing_tgz.close()

④ fetch_housing_data()

⑤ import pandas as pd
def load_housing_data(housing_path=Housing_path):
data_path = os.path.join(housing_path, "housing.csv")
return pd.read_csv(data_path)

⑥ housing = load_housing_data()

⑦ housing.head()

* Create a Test Set

① import numpy as np

② 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]
    return data.iloc[train_indices], data.iloc[test_indices]

③ train_set, test_set = split_train(data=housing)
    len(train_set), len(test_set)

O/P:- (16512, 4128)

④ from zlib import crc32
    def test_set_check(identifier, test_ratio=.2):
        total_size = 2**32
        hex_repr = crc32(np.int64(identifier)) & 0xffffffff
        in_test = hex_repr < (test_ratio * total_size)
        return in_test

⑤ def split_train_test_by_id(data, test_ratio, id_column):
    ids = data[id_column]
    in_test_set = ids.apply(lambda id: test_set_check(id, test_ratio))
    return data.loc[~in_test_set], data.loc[in_test_set]

⑥ train_set, test_set = split_train_test_by_id(data=housing_with_id,
                                test_ratio=0.2, id_column="index")
    train_set.shape, test_set.shape

This step divides data set into test and train sets with 20% of data in test dataset. The indices are selected randomly.

7) from sklearn.model_selection import

4th April, 2024.

Week - 3

* Describe & Visualize the Data to Gain Insights

① housing = strat_train_set.copy(); housing.shape

(16512, 10)

② housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()

③ housing[['population', 'median_house_value']].corr()

	population	median_house_value
population	1.000000	-0.026882
median_house_value	-0.026882	1.000000

④ housing.describe()

⑤ housing.hist(bins=50, figsize=(20,15)) \nplt.show()

The descriptive statistics of dataset are presented including mean, standard deviation and percentiles of numerical attributes

* Select & Train Model

① from sklearn.linear_model import LinearRegression

② lin_reg = LinearRegression()

③ lin_reg.fit(X=housing_prepared, y=housing_labels)

④ from sklearn.metrics import mean_squared_error

⑤ housing_predictions = lin_reg.predict(housing_prepared)

* Fine tune the model

① from sklearn.model_selection import GridSearchCV

② forest_reg = RandomForestRegressor()

③ grid_search.fit(X=housing_prepared, y=housing_labels)

GridSearchCV
estimator: RandomForestRegressor

* Launch, Monitor and maintain your system

The final model is prepared, monitoring code is written to check the system's performance. The model is retrained on a regular basis with fresh data to avoid slow degradation

4/4/2024

Week - 4

18th April, 2024.

* Linear Regression

① from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

② # Load data
data = pd.read_csv("/content/50_Startups.csv")
data = pd.get_dummies(data, columns=['State'], drop_first=True)

③ # Split data into features & target variable
~~X = data~~
Split data into training & testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

④ linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
y_pred_linear = linear_reg.predict(X_test)

⑤ mse_linear = mean_squared_error(y_test, y_pred_linear)
print("Mean Squared Error:", mse_linear)

O/P:

Mean Squared Error: 82010363.04430099

* Multiple Regression

① data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep='\s+', skiprows=22, header=None)

② data = pd.DataFrame(data)

③ X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

④ # Creating & Training Regression Model
model = MultipleRegression()
model.fit(X_train, y_train)

⑤ plt.scatter(y_test, y_pred)
plt.xlabel("True values")
plt.ylabel("Prediction")
plt.title("True values vs Prediction")
plt.show()

Week 5 25th April, 2024.

ID3 Decision Tree

① import pandas as pd
from sklearn.datasets load_iris
from sklearn.model_selection import train_test_split

② # Load the dataset
iris = ~~data~~ load_iris()
X = iris.data
y = iris.target

③ X_train, X_test, y_train, y_test = test_train_split(X, y,
test_size = 0.2, random_state = 42)

model = DecisionTree Classifier (Criterion = 'entropy', random_state = 42)

④ model.fit(X_train, y_train)

⑤ y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

O/P : Accuracy: 1.0

⑥ new_sample = [[5.1, 3.5, 1, 4, 0.2]]
Predicted_class = model.predict(new_sample)
predicted_species = iris.target_names[predicted class][0]
print("Predicted species:" predicted_species)

O/P -

Predicted specie : setosa

#play-tennis.csv dataset

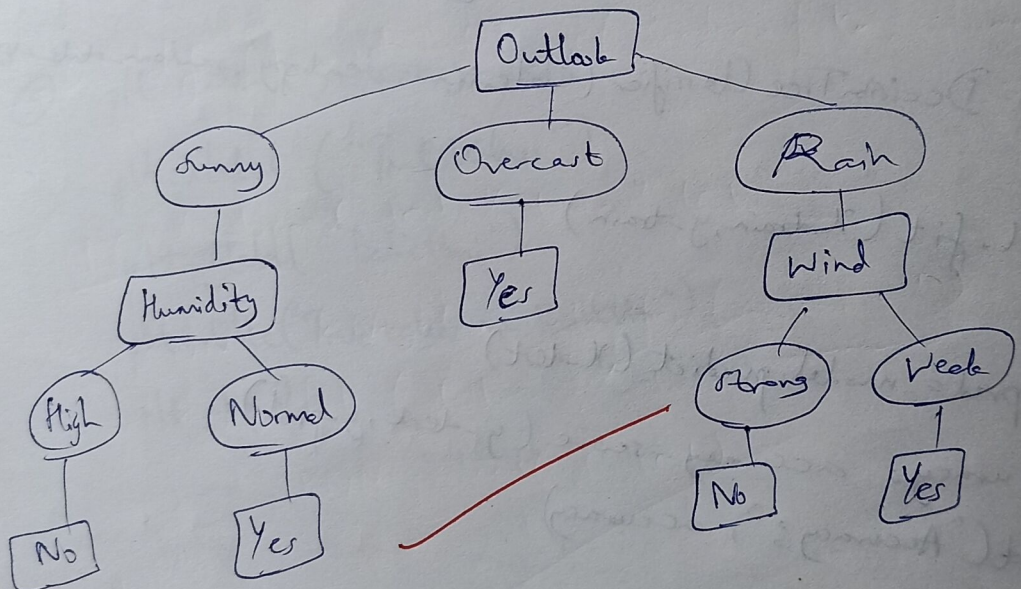
⑦ tree = buildTree(df)

⑧ import pprint

pprint.pprint(tree)

o/p:

{'outlook': {'Overcast': 'Yes',
'Rain': {'wind': {'strong': 'No', 'weak': 'Yes'}},
'Sunny': {'humidity': {'High': 'No', 'Normal': 'Yes'}}}}



25/4/2024

Week - 6 9th May, 2024.

Logistic Regression

① import numpy as np
import pandas as pd

from sklearn.preprocessing import standardscaler
from sklearn.linear_model import logistic regression

② mushroom_ds = pd.read_csv("kaggle/input/mushroom.csv")
mushroom_ds.head()

o/p

Dounts	shop	gill element	gill colour
4372	2	2	10
11461	2	2	10
11371	6	2	10

③ X = mushroom_ds([["Dounts": "shop"]])
Y = mushroom_ds("dounts")

④ x_train = scalar.fit_transform(X_train)
Y_train = scalar.fit_transform(Y_test

o/p:
accuracy > 0.6)

KNN classification :-

~~Dataset~~ :

① import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

② df = pd.read_csv ("prostate.csv")
df.head()

③ scalar = standardscalar()
scalar.fit (df.drop ("Target")
scalar_fater = scalar.transform (df.drop ('Target')

④ df_fit = pd.Dataframe (Scaler_fater, columns = df_clumn

⑤ X_train, X_test, Y_train, Y_test

⑥ KNN = KNeighbour - classifier (neighar = 1)
KNN.fit (X_train, Y_train)
pred = KNN.predict(X_test)

o/p :

[[10,3] [2,6]])

Week - 7

23rd May, 2024.

Support vector machine

```
① import pandas as pd
  from sklearn import datasets
  from sklearn.model_selection import train_test_split
  from sklearn.svm import SVC
  from sklearn.metrics import accuracy_score

② iris = datasets.load_iris()
  x = iris.data
  y = iris.target

③ x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
                                      random_state=42)

④ svm_model = SVC(kernal='linear')
  svm_model.fit(x_train, y_train)
  y_pred = svm_model.predict(x_test)

⑤ accuracy = accuracy_score(y_test, y_pred)
  print("Accuracy of svm model:", accuracy)
```

K-Means Algorithm

① import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

② data = pd.read_csv("data.csv")
X = data.values

③ kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

④ data['Cluster'] = labels
data.to_csv('clustered_data.csv', index=False)

⑤ plt.scatter(X[:,0], X[:,1], c=labels, cmap='viridis')
plt.show()

PCA

① import pandas as pd
from sklearn import datasets
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

② iris = datasets.load_iris()
X = iris.data
y = iris.target

④ pca = PCA(n_components = 2)
X_pca = pca.fit_transform(X)

⑤ pca_df = pd.DataFrame(data=X_pca, columns=['Principal Component 1', 'Principal Component 2'])
pca_df['Target'] = y

⑤ plt.figure(figsize=(8,6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], c=pca_df['Target'], cmap='viridis')
plt.colorbar()
plt.show()

Week-8

30th May - 2024.

Random Forest Ensemble Method

① import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

② iris = load_iris()
X = iris.data
y = iris.target

③ X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Build RandomForest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

④ #Predict using trained model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of Random Forest Model:", accuracy)

O/P:

Accuracy of Random Forest model: 1.0

Boosting Ensemble Algorithm

① import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

② iris = load_iris()
X = iris.data
y = iris.target

③ X_train, X_test, y_train, & y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Build AdaBoost Model using DecisionTreeClassifier as base

base_estimator = DecisionTreeClassifier(max_depth=1, random_state=42)

ada_model = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)

④ ada_model.fit(X_train, y_train)

⑤ y_pred = ada_model.predict(X_test)

⑥ accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of AdaBoost model:", accuracy)

O/P:-

Accuracy of AdaBoost model: 1.0

30/5/2024