LAB-01

1 Write a python program to import and export data using pandas library functions

importing data

```
import pandas as pd
# Read the csv file
airbnb_data = pd.read_csv("data/listings-austin.csv")
# View the first 5 rows
airbnb_data.head()
```

output no

id	name	host_id	host_name	latitude	price
2265	Zen-East	2466	paddy	30.27752	179
5245	EO-friendly	2465	paddy	30.2715	114
5456	Walk to 6th	8028	Sylvia	30.2607	129
5769	NW Austin Room	8186	Todd	30.45694	49
6413	gem of a studio	13812	Todd	30.24884	108

Reading data from URL

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
col_names = ["sepal-length_in_cm", "sepal_width in_cm", "petal_length_in_cm", "petal_width_in_cm"]
iris_data = pd.read.csv(URL_name = col_names)
iris_data.head()
```

TO Exporting data

```
df_to_csv("path/new_name.csv")
```

Week-01

Step-1:- Framing the problem & looking at the big picture

Step 2:- Get the data

```
import os
import tarfile
import urllib
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/
master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"
def fetch_housing_data(housing_url=HOUSING_URL, housing_path=
HOUSING_PATH):
fetch_housing_data()
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)
housing = load_housing_data()
housing.head()
```

step3:- Discover & visualize the data to gain insights

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

4) prepare the Data for machine learning algorithms.

```
housing = start_train_set.drop("median_house_value", axis=1)
housing_labels = start_train_set["median_house_value"].copy()
housing.shape, housing_labels.shape
housing_cat = housing[['ocean_proximity']]
housing_cat.head(10)
```

5) selecting

```
from sk
lin_reg =
lin_reg.f
some_da
some_lab
some_da
print(s
```

4/4/2020

5) Selecting & Train a model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X = housing_prepared, y = housing_labels)

some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)
print("Predictions:", lin_reg.predict(some_data_prepared))
```

4/4/2024

LAB-04

Simple linear Regression

```
def estimate_coef(x,y):
n = np.size(x)

m_x = np.mean(x)
m_y = np.mean(y)

SS_xy = np.sum(y*x) - n*m_y*m_x
SS_xx = np.sum(x*x) - n*m_x*m_x

b_1 = SS_xy/SS_xx
b_0 = m_y - b_1*m_x
return (b_0, b_1)

def plot_regression_line(x,y,b):
plt.scatter(x,y, color="m", marker="o", s=30)

y_pred = b[0] + b[1]*x

plt.plot(x, y_pred, color="g")

plt.xlabel('x')
plt.ylabel('y')

def main()
x = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([1,3,2,5,7,8,8,9,10])
b = estimate_coef(x,y)
print("estimated coeffecients:\nb=0={}\nbb1={}".
format(b)).
```

multilinear Regression

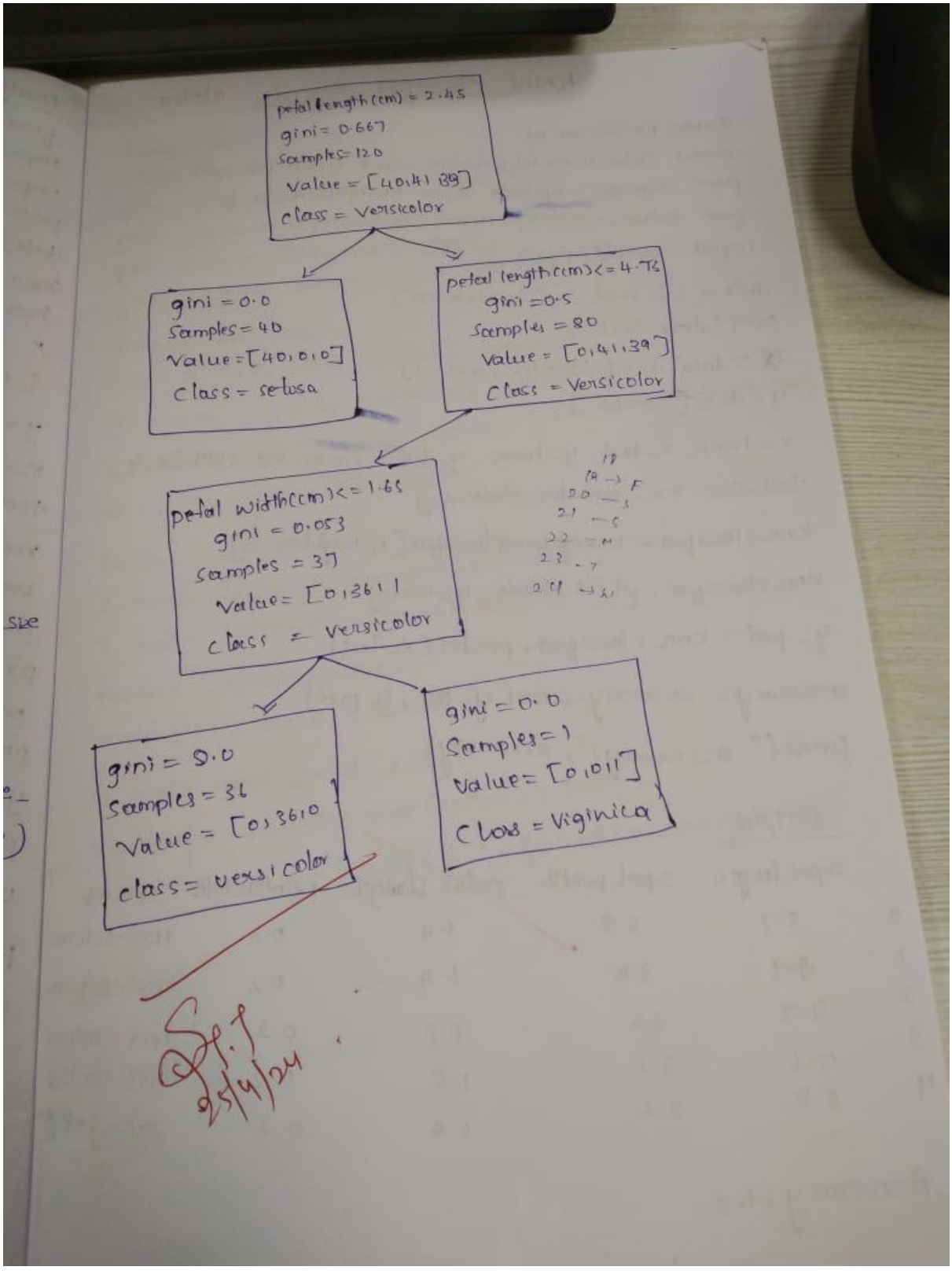
```
from sklearn model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn datasets, linear_model, metrics
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22,
header=None)
x = np.hstack([raw_df.value[::2,:], raw_df.value
[1::2,:2]])
y = raw_df.values[1::2,2]
x_train, x_test, y_train, y_test = train_test_split(x,y,
test_size = 0.4, random_state=1)
reg = linear_model.LinearRegression()
reg.fit(x_train, y_train)
print('coefficients:', reg.coef)
print('Variance score: {}'.format(reg.score(x_test, y_test))
plt.style.use('fivethirtyeight')
plt.scatter(reg.predict(x_train), reg.predict(x_train)-
y_train, color="green", s=10, label='train data')
plt.scatter(reg.predict(x_test), reg.predict(x_test)-
y_test, color="blue", s=10, label='test data')
plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)
plt.legend(loc='upper right')
plt.title("Residual errors")
plt.show()
```

Lab-05 → Decision tree 25/4/24

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
iris_data = load_iris()
iris_df = pd.DataFrame(data = iris_data, columns =
iris_data.feature_names)
iris_df['target'] = iris_data.target
print(iris_df.head())
X = iris_data.data
Y = iris_data.target
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size
=0.2, random_state=42)
dt_classifier.fit(X_train, y_train)
plt.figure(figsize=(12,8))
tree.plot_tree(dt_classifier, feature_names = iris_data, feature_
names, class_names = iris_data.target_names, filled=True)
plt.show()
```

Lab-05 → Decision tree 25/4/24

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
iris_data = load_iris()
iris_df = pd.DataFrame(data = iris_data, columns =
iris_data.feature_names)
iris_df['target'] = iris_data.target
print(iris_df.head())
X = iris_data.data
Y = iris_data.target
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size
=0.2, random_state=42)
dt_classifier.fit(X_train, y_train)
plt.figure(figsize=(12,8))
tree.plot_tree(dt_classifier, feature_names = iris_data, feature_
names, class_names = iris_data.target_names, filled=True)
plt.show()
```


petal length (cm) = 2.45
gini = 0.667
samples = 120
value = [40, 41, 39]
class = versicolor
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal length (cm) <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = versicolor
petal width (cm) <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
gini = 0.0
samples = 36
value = [0, 36, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

KNN classifier

```
import pandas as pd
import sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
data = pd.read_csv('iris.csv')
print(data.head())
X = data.drop('species', axis=1)
y = data['species']
x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(x_train, y_train)
y_pred = knn_classifier.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.4	1.4	0.2	Iris-setosa

Accuracy: 1.0

Logistic Regression

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

diabetes = load_diabetes()
x, y = diabetes.data, diabetes.target
y_binary = (y > np.median(y)).astype(int)
x_train, x_test, y_train, y_test = train_test_split(x, y_binary,
test_size=0.2, random_state = 42)

Scaler = StandardScaler()
x_train = Scaler.fit_transform(x_train)
x_test = Scaler.transform(x_test)

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

y_pred = model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%", format(accuracy * 100))
```

Output

Accuracy = 73.03%

9/5/2024

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('irisdata.csv')
print(data.head())
print(data.isnull().sum())
features = data.drop('species', axis=1)
target = data['species']
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
print(features_scaled[:5])
sse = []
for k in range(1, 11):
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(features_scaled)
sse.append(kmeans.inertia_)

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

labels = kmeans.labels_

data['cluster'] = labels
```

23/5/24

```
plt.figure(figsize=(10,6))
sns.scatterplot(x=features_scaled[:,0], y=features_scaled[:,1], hue=target, palette='viridis')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('original species visualization')
plt.show()
```

Support Vector Machine

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
data = pd.DataFrame(data = iris.data, columns = iris.feature_
names)
data['target'] = iris.target
print(data.head())
X = data.drop('target', axis=1)
Y = data['target']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.2, random_state = 42)
svm_classifier = SVC(kernel = 'linear')
svm_classifier.fit(X_train, Y_train)
Y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy :", accuracy)
```

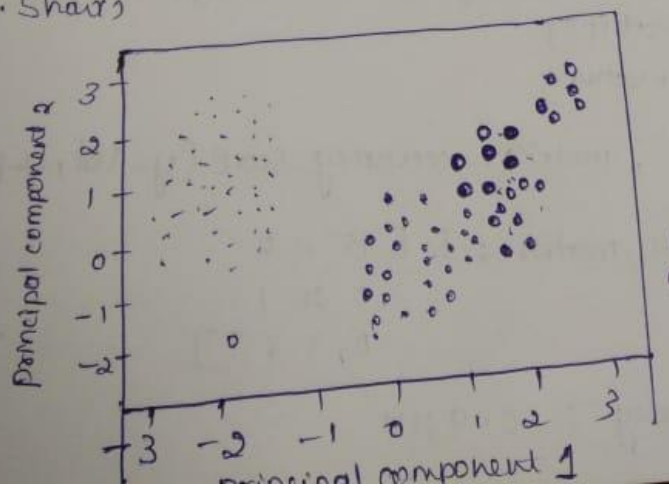
Sepal length	sepal width	petal length	petal width	target
0.2	5.1	3.5	1.4	0
0.2	4.9	3.0	1.4	0
0.2	4.7	3.2	1.3	0
0.2	4.6	3.1	1.5	0
0.2	5.0	3.6	1.4	0

23/5/2024

Accuracy = 1.0

Principal component Analysis 30/8/24

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
iris = load_iris()
X = iris.data
y = iris.target
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
df_pca = pd.DataFrame(data=X_pca, columns=['principal component 1', 'principal component 2'])
df_pca['target']=y
plt.figure(figsize=(8,6))
for target, color in zip([0,1,2], ['r','g','b']):
indices= df_pca['target']==target
plt.scatter(df_pca.loc[indices, 'principal component 1'],
df_pca.loc[indices, 'principal component 2'], c=color, s=50)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.legend(iris.target_names)
plt.title('PCA of IRIS dataset')
plt.show()
```



Random Forest Ensemble method.

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris.target_names)
print(iris.feature_names)
print(iris.data[0:5])
import pandas as pd
data = pd.DataFrame({
    'Sepal length' : iris.data[:,0],
    'Sepal width' : iris.data[:,1],
    'petal length' : iris.data[:,2],
    'petal width' : iris.data[:,3],
    'Species' : iris.target})
from sklearn.model_selection import train_test_split
X = data[['Sepal length', 'Sepal width', 'petal length', 'petal width']]
y = data['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
print("confusion matrix")
print(confusion_matrix)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

Output:- confusion matrix : [[15 0 0
0 20 1
0 3 6]]

Accuracy : 0.91111

Boosting Ensemble Learning Method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.metrics import accuracy_score
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
adaboost_clf = AdaBoostClassifier(n_estimators=50, random_
state=42)
adaboost_clf.fit(X_train, y_train)
y_pred = adaboost_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output

Accuracy: 1.0

30/5/2024