INDEX

NAME: Sagar (M) STD.: ____ SEC.: ____ ROLL NO.: ____ SUB.: ML - Lab

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	21-3-24	lab 1		Sp.T
2	28-3-24	lab 2		Sp.T.
3	4-4-24	Lab 3		Sp.T
4	18-4-24	lab 4		Sp.T.
5	25-4-24	Lab 5		Sp.T.
6	9-5-24	Lab 6		Sp.T.
7	23-5-24	lab 7		Sp.T.
8	30-5-24	Lab 8		Sp.T.

classmate
Date 21-3-24
Page

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

1) Reading data from CSV file

```
import pandas as pd
airbnb_data = pd.read_csv("/content/sample_data/
              california_housing_test.csv")
airbnb_data.head()
```

output :-

	longitude	latitude	housing_median_age	total_rooms
0	-122.05	37.37	27.0	3885.0

total_bedrooms	population	household	median	median house value
661.0	1537.0	606.0		6.6085

2) Reading ~~& writing~~ data from URL

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data"
col_names = ["sepal-length', 'sepal width', 'petal length',
            'petal width', 'class']
iris_data = pd.read_csv(url, names=col_names)
iris_data.head()
```

3) Exporting dataframes to a CSV file

```
iris_data.to_csv("cleaned iris data.csv")
```

Output

	sepal_length	speal_width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa

5.1
2 1 3 1 2 4

Week - 2

2 End to End Machine learning Project

Get the Data

```
import os
import tarfile
import urllib

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

def fetch_housing_data(housing_data = 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()

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()

housing.info()

housing['ocean_proximity'].value_counts()

housing.describe()

import matplotlib.pyplot as plt
import seaborn as sns

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

Create a 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)
    return data.iloc[train_indices], data.iloc[test_indic

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

Sol
28/3/24

classmate
Date 4-4-24
Page

lab - 3

* Discover & Visualize the Data to Gain Insights

Visualizing the Geographical Data

```
housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()
housing.plot(kind='scatter', x='longitude', y='latitude',
alpha=0.4, s=housing['population']/100, label='population',
figsize=(10,7), c='median_house_value',
cmap=plt.get_cmap(name='jet'), colorbar=True)
plt.legend()
```

Looking for correlations

```
corr_matrix = housing.corr()
```

Prepare the Data for Machine learning Algorithm

```
housing = strat_train_set.drop("median_house_value")
housing_labels = strat_train_set["median_house_value"].
                 copy()
```

* Data cleaning

```
housing.dropna(subset='total_bedrooms')
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop("ocean_proximity")
imputer.fit(housing_num)
```

Handling text & categorical Attributes

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

* Select & Train a Model

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

tree_reg = DecisionTreeRegressor

forest_rmse = RandomForestRegressor()

* Fine-Tune your Model

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(estimator = forest_reg, param_grid = param_grid, scoring = 'neg_mean_square error', cv = 5, return_train_score = True, n_jobs

* ~~Lean~~ Launch, Monitor, & Maintain your Syst

We can hot-load the model with in web application or alternatively, wrap the model around its own API end-point and design web component seperately

classmate
Date 18-4-2024
Page

lab - 4
Simple Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coeff(x, y)
    n = np.size(x)
    # mean of x, y vectors
    # calculating cross-deviation & deviation about x
    # calculating regression coefficient

def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = 'm', marker = 'o', s = 30)
    # plotting the reg line
    plt.show()

def main():
    #observation / data
    # estimating coefficient
    plot reg line

if __name__ == "__main__":
    main()
```

Multiple Linear Regression

```
Import sklear.model_selection import
train_test_split
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import dataset, linear_model,
    metrics
```

* Import data from url
* reg = linear_model.LinearRegression()
* calculate coefficient
* calculate variance
* plot the Residual errors

Sol
18/4/22.

Lab 5

Decision tree ID3 Algorithm Implemention

Steps

* Reading the data sets
* Importing packages numpy, pandas
* Reading the data set and set the indexes

Decision tree

ID3 is simple decision tree algorithm

Emplement used functions

$$\text{Entropy} = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Average Information

$$I = \sum \frac{p_i + n_i}{p+n} \text{Entropy(attribute)}$$

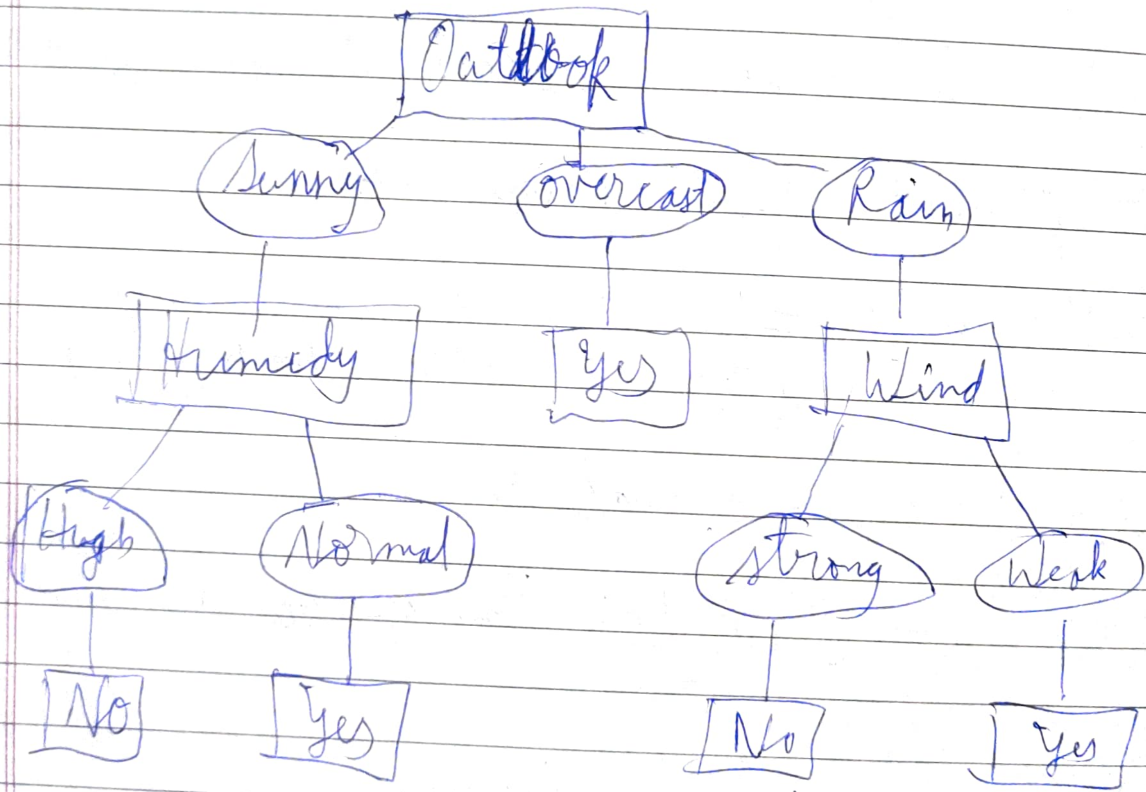
Information Gain

$$\text{Gain} = \text{Entropy}(S) - I_{(Att)}$$

Build Decision tree

* build Tree function construct a decision tree recursively
* select the attribute with high information gain
* process continue till subsets are pure

pprint the tree



25/4/24

classmate
Date 1-5-24
Page

Lab 6

Logistic Regression

Key concept

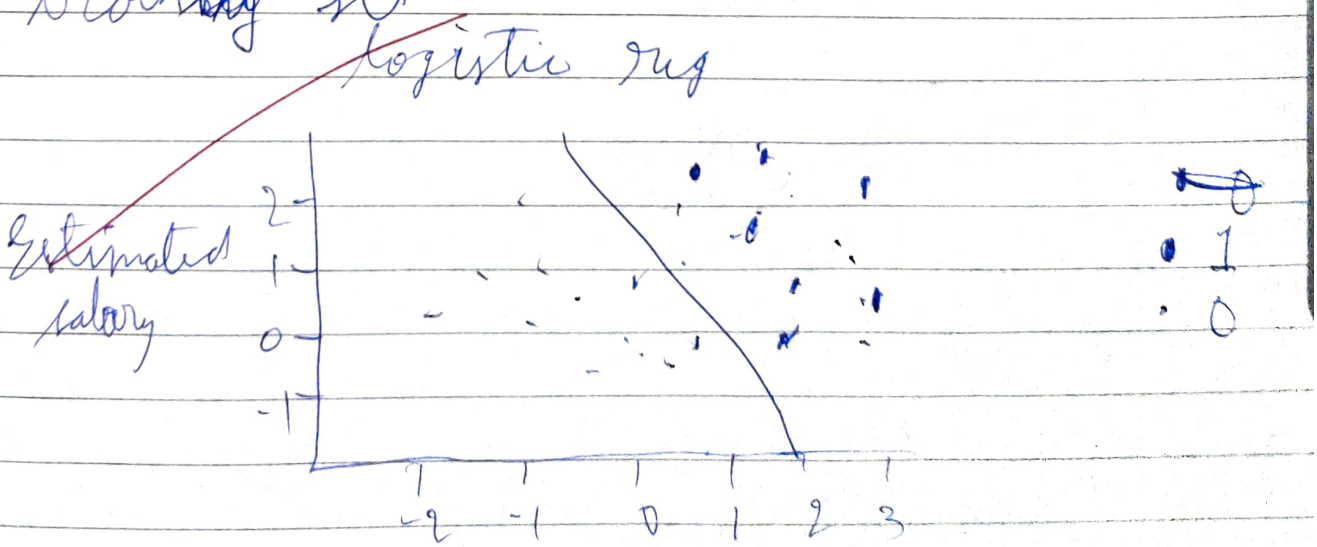
Sigmoid function :- It is a main function that ensures output are between 0 & 1 by converting a linear combination of input data into probabilities

$$\sigma(z) = \frac{1}{1+e^{z}}$$

where z is a linear equation.

Steps

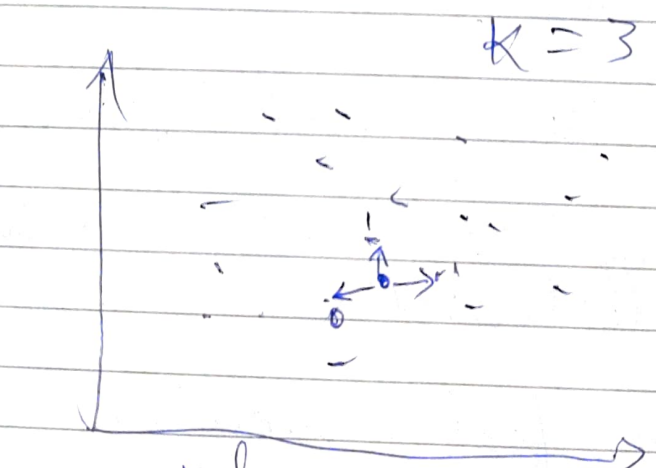
* Import the dataset from csv file
* splitting the dataset into training and test set
* splitting feature scaling
* fitting the logistic regression into the training set



KNN Model

Key concept

* select the right K value
* train the model by feeding the training dataset
* Slyt the test set value, and find the Manhattan distance between the points
* select the min value of K data
* majority value containing will be the answer



KNN

Sept 9/8/24

classmate
Date 23-5-24
Page

Week-7

Support vector Machine

```
import pandas as pd
from sklearn import dataset
from sklearn import *

Iris = dataset.load_iris()
x = iris.data
y = iris.target

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

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

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

K-means Algorithm

1) import matplotlib.pyplot as plt
from sklearn import *

2) data = pd.read_csv("data1.csv")
x = data.values

3) Kmeans = KMeans(n_clusters=3, random_state=0)
Kmeans.fit(x)

4) data['cluster'] = label
data.to_csv('cluster_data.csv', index = False)

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

PCA

1) import pandas as pd
import sklearn as sk

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

3) pca = PCA(n_components = 2)
x_pca = pca.fit_transform(x)

4) 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()

Sol:1
23/8/24

classmate
Date 30-5-24
Page

lab 8

~~Random~~ ~~Forest~~ Boosting Implementation

```
import pandas as pd
from sklearn import *

iris = load_iris()
X = iris.data
Y = iris.target

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

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

ada_model.fit(X_train, y_train)

y_pred = ada_model.predict(X_train)

acc = accuracy_score(y_test, y_pred)
```

Random Forest Implementation

```
import pandas as pd
from sklearn import *

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

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

rf = RandomForestClassifier(n_estimators=100,
                            random_state=42)

rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

ac = accuracy_score(y_test, y_pred)
```