LAB - 1

① Reading data from URL

Code:

```
import pandas as pd
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", "class"]
iris_data = pd.read_csv(url, names = col_names)
iris_data.head()
```

Output:

	Sepal_length_in_cm	Sepal_width_in_cm	petal_length_in_cm
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7		

petal_width_in_cm	class
0.2	Iris-setosa
0.2	Iris-setosa

② Reading data from csv file

```
import pandas as pd
data = pd.read_csv("iris_data.csv")
data.head()
```

OUTPUT:

	Sepal_length_in_cm	Sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm
0	5.1	3.5	1.4	0.2

class
Iris-setosa

LAB - 2

1. Look at the big picture.
   - The features and columns of dataset must be determined.
   - Frame the problem statement and questions which will be addressed in the work flow. In this problem regression task is considered
   - Select the performance measure.
     In regression problem root mean squared error is taken as performance measure.
     If there are many outliers mean absolute error is used.

2. Get the data.
   It is preferable to create some util functions to automate the process of downloading / loading used based data sets.
   Each of URL are extracted into housing
   Histogram for all numerical data points are plotted to count the count.
   Test Data set is sampled from the above (20%) using 80% for training.

LAB-3

Create Data Set.
80% of the data set is taken as training set and 20% is taken as testing set.
Considering test data if program is run again, it generates different test set.
The test data are always preferred to be hidden. It is scalable and extendable.

3. Discover & Visualize the Data.
For visualising training set is analysed it is preferred to make copy of training set.
Examining dataset since we have latitude and longitude information, map visualisation is done.
Various visualizations parameters like colour, alpha value, figsize, kind are used to make the visualisation look better.

Looking for Correlation.
Various attributes will be inter related or vary with each other in a pattern, this is give by correlation. Correlation coefficient lies between -1 to 1. When the coefficient is close to 1 it means there is strong positive correlation the variable and vice versa -1.

4. Prepare Data for ML Algorithms.
-Data cleaning.
It is the initial step where the missing values, null values etc are handled. Some

data which are not in format are also removed transformed. Null values can be removed or filled using several methods like backward fill, forward fill or Interpolation.

Handling Text & Categorical Attributes.
Dataset has only 1 categorical data. Grouping can be used for this.

Feature Scaling.
ML algos don't perform well on data with different scales. The 2 ways of scaling are:
i) Min Max scaling.
ii) Standarization

5. Select and Train Model.
The selected and cleaned data is used to train model. First Linear Regression Model is used on the training dataset.
The overfitting and underfitting is handled during training.
RMSE model and decision tree model is applied.
RMSE model fits better. The model is saved after traing so that it can be used anytime. joblib is used for storing.

6. Fine Tune Your Model.
Scikit-learn's GridSearchCV is used for this purpose. Along with this RandomizedSearchCV is used for better purpose.

Date
Page 18/9/24

Calculator Q: LAB-4

Linear Regression.

1) Simple linear regression.
It involves predicting a dependent variable based on single independent variable.

2) Multiple linear regression.
Involves predicting a dependent variable based on multiple independent variable.

$$y = mx + c$$

y → dependent, x → independent.

Code:

```
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_reg_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, 4, 3, 9, 10, 12, 14])

    b = estimate_coef(x, y)
print("Estimated coef: \nb_0 = {} \
      \nb_1 = {}", format).
```

Date
Page 25/4/24

Lab - 5.

Decision Tree - ID3.

```
import numpy as np
import pandas as pd

    eps = np.finfo(float).eps
    from numpy import log2 as log

import os

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dir, file))
df = pd.read_csv('path')
df = df.drop ['day', axis=1]
df.head()

print("Rows: {df.shape[0]}, columns: {df.shape[1]}")
print(df.columns)
df.info()
df.describe()
```

Handling missing data

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

(i) We compute entropy for dataset.
(ii) Take average info entropy for current attribute.
(iii) Calculate gain.

(iv) Pick highest gain attribute.
(v) Repeat until we get desired tree.

Average information:

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

Information gain:

Gain = Entropy(S) - I(attribute)

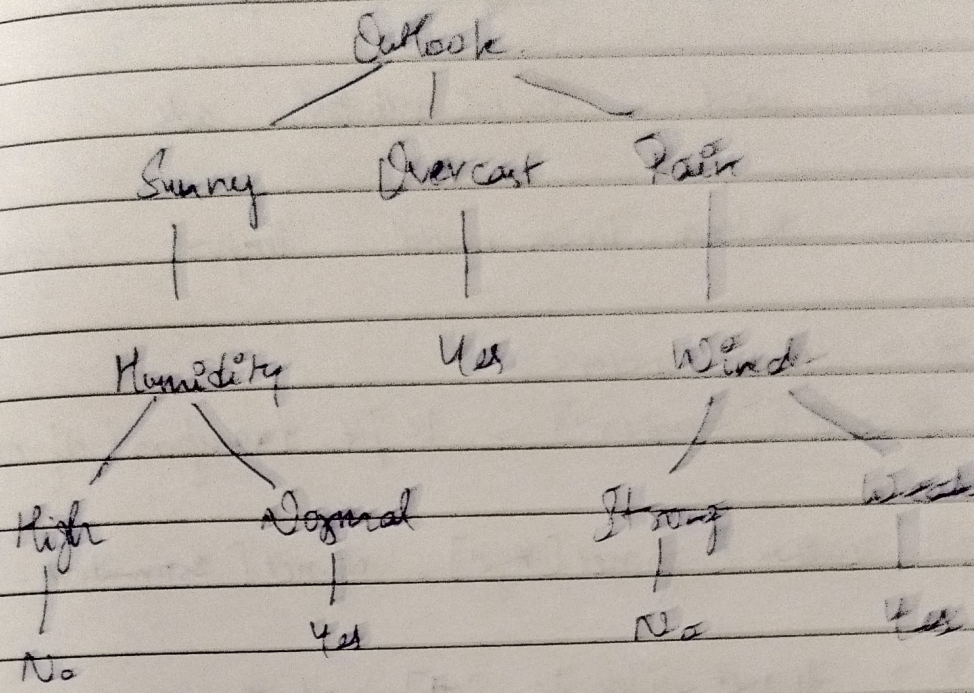
Building Decision Tree

```
def buildTree(df, tree=None):
    target = df.keys()[-1]
    if tree is None:
        tree = {}
        tree[node] = {}

    for value in att value:
        subtable = get_subtable(df, node_value)
        if len(counts) == 1:
            tree[node][value] = cl value[0]
        else
            tree[node][value] = buildTree(subtable)
    return tree

tree = buildTree(df)
pprint.pprint(tree)
```

OUTPUT



Code Output,

{'Outlook': {'Overcast': 'Yes', 'Rain': {'Strong': No, 'Weak': 'Yes'}, 'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}

Lab - 6.

Logistic Regression and KNN.

Dataset used: Social_network_ads

```
from sklearn.linear_model import LogisticRegression

le = LabelEncoder()
df_net['Gender'] = le.fit_transform(df_net['Gender'])

plt.scatter(df_net['Age'], df_net['Estimated Salary'])

X = df_net.iloc[:, :-1].values
Y = df_net.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split
                    (X, Y, test_size=0.25, random_state=1)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

accuracy_score(y_test, y_pred)
=> 0.83.

print("f1 score: {f1_score(y_test, y_pred)}")
=> f1 score: 0.78

cf_matrix = confusion_matrix(y_test, y_pred)
```

sns.heatmap(cf_matrix)

for KNN, till feature scaling procedure is same.

from sklearn.neighbours import KNeighboursClassifier as KNN

classifier = KNN(n_neighbours = 5, metric = 'minkowski', p = 2)

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

accuracy = score(y_test, y_pred)
⇒ 0.88

print("F1 score = {f1_score(y_test, y_pred)}")
= F1 score = 0.86

cf_matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(cf_matrix)

for sample data,
logistic regression, age = 45, salary = 97000
predicted result = 1 (purchased ads)

KNN, Age = 30, Salary = 87000
predicted result = 0 (not purchased)

Correlation matrix.

for LR:

y_pred \ y_true	0	1
0	52	6
1	11	31

y_true

for KNN:

y_pred \ y_true	0	1
0	50	8
1	4	38

y_true.

0,0 → True negative.
0,1 → False positive
1,0 → False negative
1,1 → True negative.

Lab - 7

SVM

```
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

cancer = load_breast_cancer()
    x = cancer.data[:, :7]
    y = cancer.target

svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
svm.fit(x, y)

DecisionBoundaryDisplay.from_estimator(
        svm, x, response_method="predict",
        cmap = plt.cm.spectral,
        alpha = 0.8,
        xlabel = cancer.feature_name[0],
        ylabel = cancer.feature_name[1],
    )
plt.scatter(X=[:0], x[:,1], c=y, s=20, edgecolor)
plt.show()
```

PCA

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
    data = pd.read_csv("...")
    scaling = StandardScaler()
    scaling.fit(data)
```

```
principal = PCA (n_component = 3)
principal.fit (scaled_data)
X = principal.transform (scaled_data)

plt.figure (fig size = (10, 10))
plt.scatter (x [:0], x [:1], c = data['target'],
                                cmap = 'plasma')

plt.xlabel ('PC1')
plt.ylabel ('PC2')
```

K - Means:

```
import pandas as pd
  data = pd.read_csv (" _ _ ")
  data.head ()
import numpy as np
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.cluster import kmeans
  x, y = load_iris (return_x_y = True)
  kmeans = kmeans (n_clusters = 3, random_state = 0)
  kmeans.fit (X)
  pred = kmeans.fit_predict (X)
  pred.
```

Date
Page 30/9/24

Week - 8.

Ensemble Random forest and ADA Boost

```
from sklearn.datasets import make_moons
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,
                             AdaBoostClassifier, GradientBoostingClassifier

X, y = make_moons (n_samples = 10000, noise = .5,
                   random_state = 0)

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

clf = DecisionTreeClassifier ()
clf.fit (X_train, y_train)
y_pred = clf.predict (X_test)
accuracy_score (y_test, y_pred)
```

=> 0.753.

```
clf = RandomForestClassifier (n_estimators = 100, max_features = 'auto',
                              random_state = 0)
clf.fit (X_train, y_train)
y_pred = clf.predict (X_test)
accuracy_score (y_test, y_pred)
```

=> 0.7965

```
clf = AdaBoostClassifier (n_estimators = 100)
clf.fit (X_train, y_train)
y_pred = clf.predict (X_test)
accuracy_score (y_test, y_pred)
```

=> 0.7965.

clf = AdaBoostClassifier(n_estimators = 100)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
accuracy_score(y_test, y_pred)

=> 0.833.

Sol:
30/8/24