INDEX

Name : Rachana.R Class : .....

Section : ..... Roll No. : ..... Subject : ML Lab.

Sl. No.	Date	Title	Page No.	Teacher Sign. / Remarks
	21/3	Importing & Exporting.		
	28/3	End to End project		
	4/4	Linear & Multiple regression.		
	18/4	Decesion Tree - ID3.		
	25/4	KNN & ~~Logistic~~		
	9/5	Logistic Regression.		
	23/5	K Means.		
	23/5	SVM		
	23/5	PCA		
	30/5	Random Forest & Ada Boosting		

Week - 01

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

Code:

```
import pandas as pd
iris_data = pd.read_csv("iris.csv")
iris_data.head()
iris.columns = col_names.
```

O/p

	sl	sw	pl	pw	class
0	5.1	3.5	1.4	0.2	iris-setosa
1	4.9	3.0	1.3	0.2	iris-setosa
2	4.7	3.2	1.4	0.2	iris-setosa
3	4.6	3.1	1.5	0.2	iris-setosa
4	5.0	3.6	1.4	0.4	iris-setosa

Code:

```
url = "https://archive.ics.uci.edu/ml/machine
        Learning-database/
col_names = ["sepal_length_in_cm",
             "sepal_width_in_cm",
             "petal_length_in_cm",
             "petal_length_in_cm"]
iris_data = pd.read_csv(url, name=col_name
iris_data.head()
```

Output:

	sl	sw	pl	pw
0	5.1	3.5	1.4	0.2

```
iris_data.to_csv("cleaned_iris_data.csv")
```

Lab -2

Get the Data

→ Download the data

```
import os
import tarfile
import urllib
```

→ Using these libraries, download and extract housing_data and load the data into "housing.csv".

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

→ Using the above commands inspect the attributes of the loaded housing dataset.

→ Using matplotlib and seaborn libraries, by plotting the hisplot detect the outliers.

Create the Test set

→ Splitting the data set on test ratio 0.2, i.e, training data is 80% of dataset and testing data is 20% of dataset

→ Stratified sampling is when random chosen data are representative of a whole target population. Each homogeneous subgroup is called strat

Lab - 03.

Discover and visualize the data to gain insights.

Visualize the data using matplotl and seaborn libraries.

Calculating the standard correlation coefficient of every pair of colums.

Prepare the data for machine learning algorithms.

Data cleaning, handling text and categorical data, custom transformer, feature scaling, transformation pipelines etc are done here.

Select and Train model

At first linear regression model is used to train but the model is overfitting the data

To tackle this, Decesion Tree Regressa model is used as it is capable of finding non-linear relationships with data. But DTM is also overfitting so badly that it perfoms worse than the linear RM.

Fine tune your model.

At last fine tuning of model is carried out, evaluating on the test set and then deploy, monito and maintaining the system.

Lab-04

Python implementation of linear regression

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

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')
    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, ..., 9])
    y = np.array([1, 3, 2, ..., 12])
    b = estimate_coef(x, y)
    print(b)
    plot_regression_line(x, y, b)
```

O/P

(b0, b1) = (1.263, 1.1669...)

→ Multiple linear regression.

```
from sklearn.model_selection import
    train_test_split
import numpy as np
from sklearn import datasets,
    linear_model, metrics

data_url = "
raw_df = pd.read_csv
x = np.hstack ([raw_df.values[:2, :]
    raw_df values [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 ("co-efficients:", 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, colour = "green",
    s=10, label = "Train data)
```

```
plt.scatter(reg.predict(x_test),
    reg.predict(x_test) - y_test,
    colour = "blue", s=10,
    label = "test data")
plt.lines(y=0, xmin=0, xmax=50,
    linewidth=2)
plt.legend(loc='upper right')
plt.title("residual errors")
plt.show()
```

Lab-05

ID3

```
import numpy as np
import pandas as pd.
eps = np.finfo(float).eps.
from google.colab import drive
drive.mount('/content/drive')
Path = 'drive/My Drive/ML lab/datasets/
       play_tennis.csv'
df. pd.read_csv(Path)
df
```

```
def find_entropy(df):
    target = df.keys()[-1]
    entropy = 0
    values = df.[target].unique()
    for value in values:
    fraction = df[target].value_counts()
        [value] / len(df[target])
    entropy += -fraction * np.log2(fraction)
    return entropy
def average information [df, attribute]
    target = df.keys()[-1].
    target_variables = df[target].unique()
    variables = df[attribute].unique()
    entropy2 = 0.
```

```
for variable in variables:
    entropy = 0
    for target_variable in target_variables:
        num = len(df[df[attribute] == variable][df[target] == target_var])
        den = len(df[df[attribute] == variable])
        fraction = num / (den + eps)
        entropy += -fraction * log(fraction + eps)
    fraction2 = den / len(df)
    entropy2 += -fraction2 * entropy
    return abs(entropy2)
```

KNN Implementation

```
import numpy as np
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
dataset = pd.read_csv('/content/
        drive/My Drive/Iris_csv')
dataset.head()
dataset.groupby('species').size()
    species
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
feature_columns = ['sepal_length',
    'sepal_width', 'petal_length',
    'petal_width']
X = dataset[feature_columns].values
y = dataset['species'].values
from sklearn.preprocessing import
    LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
from sklearn.model_selection
    import train_test_split
X_train, X_test, y_train, y_test
    = train_test_split(X, y)
```

```
import matplotlib.pyplot as plt
import seaborn as sns.
%matplotlib inline
from sklearn.neighbors import
    kNeighborsClassifier
from sklearn import confusion_matrix
from sklearn.model_selection
    import cross_val_score
classifier = kNeighborsClassifier(n=3)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
accuracy = accuracy_score(y_test,
    y_pred) * 100.
```

O/P : Accuracy of our Model is equal 96.67%.

Lab - 7.

K-Means Implementation.

```
import pandas as pd
data = pd.read_csv("Iris.csv")
data.head()
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import
    Iris.data
from sklearn.cluster import kMean
X, Y = load_iris(return_X_y=True)
kmeans = KMeans(n_clusters=3,
random_state=2)
kMeans.fit(X)
pred = kmeans.fit_predict(X)
pred
```

Lab - 9.

SVM.

```
from sklearn.datasets import
    load_breast_cancer
from sklearn.inspection import
    DecesionBoundaryDisplay
from sklearn.svm import SVC
cancer = load_breast_cancer()
X = cancer.data[:, :2]
Y = cancer.target
svm = SVC(kernel = 'rbf', gama = 0.5)
svm.fit = (X, Y)
DecisionBoundaryDisplay.from_
    estimator(
    svm,
    X,
    response_method = 'predict',
    cmap = plt.cm.spectral,
    alpha = 0.8,
    xlabel = cancer.feature_names[0],
    ylabel = cancer.feature_names[1],
)
plt.scatter(X[:, 0], X[:, 1], c=y, s=20,
plt.show().
```

Lab -7.

PCA.

```
import pandas as pd.
import numpy as np.
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.decomposition
    import PCA.
from sklearn.preprocessing import
    StandardScaler
from sklearn.datasets import
    load_breast_cancer()
data.keys()
print [data['feature_names]).
df1 = pd.DataFrame(data['data'],
    columns = data['feature_names'])
plt.figure(figsize=(10,10))
plt.scatter(x[:,0], x[:,1], c=data:
    [target], cmap='plasma')
plt.xlabel('PCA')
plt.ylabel('PCA')
```

Lab 8

Random Forest

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import datasets
from sklearn.model_selection import
    train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
iris = datasets.load_iris()
iris_data = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_data.head()
X = iris_data.iloc[:, :-1].values
y = iris_data.iloc[:, -1].values
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.33)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
model1 = RandomForestClassifier(n=10)
model1.fit(X_train, y_train)
```

Ada Boosting

```
from sklearn.datasets import load_iris
from sklearn.model_selection import
    train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

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

X_train, X_test, y_train, y_test = train_
    test_split(X, y, test_size=0.4)

adaboost_clf = AdaBoostClassifier(n=30,
    learning_rate=1, random_state=7,2)
adaboost_clf.fit(X_train, y_train)
y_pred = adaboost_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:")
```

O/p accuracy = 0.966