


INDEX

Name ROHAN SATHISH KUMAR Sec 10

Roll No MLLAB Subject MLLAB School/College MLLAB

School/College Tel. No. Parents Tel. No.

Sl No.	Date	Title	Page No.	Teacher Sign / Remarks
1	21/3/24	Import/Export using Pandas		
2	28/3/24	2nd to 2nd ML		
3	4/4/24	Descriptive & Visualization Data		
4	18/4/24	Linear & Multiple Regression		
5	25/4/24	IO 3 Decision Tree		
6	4/5/24	Logistic Regression & RNN Classification		
7	23/5/24	SVM, K Means & PCA		
8	30/5/24	Random Forest & Boosting Ensemble Methods.		


 31.05.24

21/6 Lab-1

a) write a python program to import & export
csv file using pandas

A) import pandas as pd

df = pd.read_csv("iris.csv")

df.head()

OUT: id sepal length cm sepal width cm

3.5

petal length cm

5.1

1.4

Petal width cm
0.4

species
setosa

url = 'https://...'

col_names = ["length in cm", "width in cm",

'petal length in cm', 'petal width in cm',

'class']

data = pd.read_csv(url, names = col_names)

~~data.head()~~

OUT: ~~sepal length in cm~~

5.1

sepal width in cm

3.5

petal length in cm

1.4

petal width in cm

0.4

class

user - satish

data to use ('Houses - age') / report

28/9

Lab 2

Housing Data

Main steps involved are :-

1) Defining the problem

2) Importing the data

3) Cleaning of data, analysis of data

4) Selecting the ML Model

5) Training the Model

The data has been imported via the web

2) Cleaning & Descriptions

To describe the data set, we use the describe & head functions

Another method can be used to get info on the dataset

The dataset has 10 attributes and is 640x10 in size. The null values in each column can be found out using isnull() method to get count of NaN values.

The data types per column is also visible

Visualization :-

By making use of Matplotlib lib library histogram on each column can be drawn to display the continuous data

7 very example :-

Housing . hist (bins = 50, figsize = (10, 10))
plt . show ()

This makes use of hist function from seaborn library

4) Cleaning Test data :-

shuffled_index = np.random.permutation
(len(data))

test_size = int (len(data) * test_size)

test_index = shuffled_index [: test_size]

train_index = shuffled_index [test_size :]

return data.iloc [train_index], data.iloc
[test_index]

This divides the data set into test & train sets with 20% of data in test dataset

The index are selected randomly

5) Fit the your Model

Grid Search

from sklearn.model_selection import GridSearchCV

param_grid = [

{ 'n_estimators': [3, 10, 50] , 'Max-features':
[2, 4, 6, 8] }

{ 'bootstrap': [False] , 'n_estimators': [2, 10]

Max-features': [2, 3, 4] }

}

model = RandomForestRegressor ()

grid = GridSearchCV

(model, param_grid, cv = 5)

best_model = grid.best_estimator_

score = best_model.score (X_test, y_test)

print (best_model.score (X_test, y_test))

6. Launch, Monitor & Maintain your system.

2. Lab 4 :-

Simple Linear Regression

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from pandas.core.computation import where

Import data

df_sal = pd.read_csv('data/Salary_data.csv')

df_sal.head()

Analyze Data

df_sal.describe()

Distribution

plt.title('Salary Distribution plot')

sns.distplot(df_sal['Salary'])

plt.show()

Relationship between Salary & Experience

plt.scatter(df_sal['Years Experience'])

plt.title('Salary vs Experience')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.legend() plt.show()

Split data

Split into independent / Dependent variables

X = df_sal.iloc[:, 1]

Y = df_sal.iloc[:, 2]

Split into Train / Test sets

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

Train Model

regressor = LinearRegression()

regressor.fit(X_train, Y_train)

Predict results

Y_pred_test = regressor.predict(X_test)

Y_pred_train = regressor.predict(X_train)

Prediction on training set

plt.scatter(X_train, Y_train, color='lightblue')

plt.plot(X_train, Y_pred_train, color='red')

plt.title('Salary vs Experience')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

Confusion and Entropy

With (P, Lockbit: {regless, lock} - Y⁺)
and (P, Entail: {regless, entail} - Y)

Confusion: $\begin{bmatrix} 9312 & 5751 & 2673 \end{bmatrix}$
Entail: $\begin{bmatrix} 26750 & 099 & 51 & 063 \end{bmatrix}$

Sp. 8
18/11/24

Lab 5

Decision ID3 Algorithm Implementation

Steps

- + Reading the data sets
- + Mapping features to numeric values
- + Reading the data set & set the indices

Decision Tree

ID3 is simple decision tree algorithm

Implement used functions

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

Information Gain

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

Information gain

$$\text{Gain} = \text{Entropy}(S) - I(\text{att})$$

Build Decision tree

* Build Tree function construct a decision tree recursively

* select the attribute with high information gain

* Repeat again till subsets are pure.

print the tree :-



Sum

Logistic Regression

Key Concept :-

Sigmoid function :- It is a non-linear function that crosses output 0 or 1 by converting a linear combination of input data into probabilities.

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

where z is a linear equation.

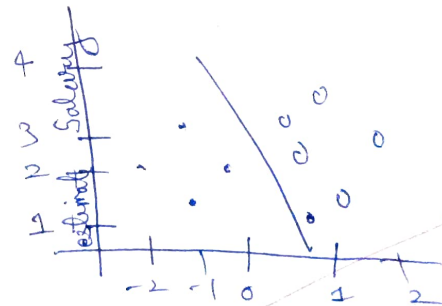
Step s:-

1 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

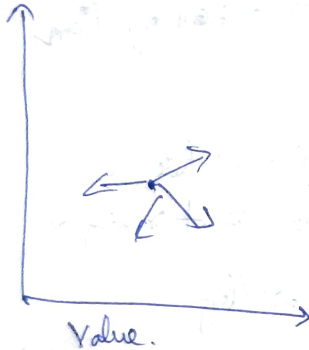


Sum

KNN

Key Concept

- * Select the right K values
- * Train the model
- * Select the test set value & find the distance
- * Select the min value of K dataset
- * Major only value containing will be the answer



Input pandas pd
input numpy int. numpy as plt
input numpy as np

① `df = pd.read_csv('input.csv')`
`df.head()`

② `train = StandardScaler()`
`Scaler.fit(df.drop('Target'))`

`Scaler.fit = scaler (train (df.drop('Target'))`

③ `df_fit = pd.DataFrame (columns=df.columns)`

④ `X_train, X_test, y_train, y_test`

⑤ `KNN = KNeighborsClassifier (n_neighbors=1)`
`KNN.fit (X_train, X_test)`

O/P

~~`[(10, 3), (5, 6)]`~~

Sam

Week 7

① Build a Support Vector Machine Model for a given dataset

import pandas as pd

from sklearn import datasets

from sklearn.model_selection import train_test_split

from sklearn import svm

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 = 0.2, random_state = 42)

Svm_model = svm.SVC(kernel = '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")

② Build K-Means algorithm to cluster a set of data stored in a csv file

import pandas as pd

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

data = pd.read_csv('data/scrub_data.csv')

cl = KMeans(n_clusters = 3)

X = data.values

KMeans = KMeans(n_clusters = 3, random_state = 42)

KMeans.fit(X)

labels = KMeans.labels

data['cluster'] = labels

data.to_csv('cluster_data.csv', index = False)

plt.scatter(X[:, 0], X[:, 1],

c = labels, cmap = 'iris')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('K-Means Clustering')

plt.show()

② Principal Component Analysis

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

③ pca = PCA(n_components=2)

X_pca = pca.fit_transform(X)

④ pca = df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'],
 index=iris.index, 'Principal Component 1')
pca = df[['Target']] = Y

⑤ plt.figure(figsize=(8,6))

plt.scatter(pca[['PC1']], pca[['PC2']])

c = pca[['Target']],
 cmap = 'irdb'

plt.colorbar()

plt.show()

Q

Week-8

1) Random Forest Ensemble Method

① import pandas as pd

data sklearn.ensemble import RandomForestClassifier
data 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)

Build Random Forest Model

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

Train the model

rf_model.fit(X_train, y_train)

④ # Predict using train model

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy of Random Forest Model:",
accuracy)

O/P

Accuracy of Random Forest : 1.0

Boosting Ensemble Algorithm

① import pandas as pd

data sklearn.ensemble import AdaBoostClassifier
data sklearn.tree import DecisionTreeClassifier
data 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 Ada Boost Model using Decision Tree

base_estimator = DecisionTreeClassifier(max_depth=3,
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 Ada Boost Model :"
accuracy)

O/P:-

Accuracy of Ada Boost Model : 1.0

Scanned with
CamScanner