

A decorative banner featuring five cards with rounded corners, each containing a large, bold, black letter. The letters are arranged horizontally and overlap slightly. The first card contains 'I', the second 'N', the third 'D', the fourth 'E', and the fifth 'X'. The background of the banner is a light beige color.

NAME : Parisha. Gotadke STD : 6th sem SEC : D ROLL NO : 1BM121CS229

21/03/2024

Week-1

Import and Export a CSV file with pandas in python

Code:

```
import pandas as pd
df = pd.read_csv("file-path.csv")
df.head()
```

Output:

Id	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.0	1.4	0.2	Iris-setosa
6	5.4	3.9	1.5	0.2	Iris-setosa
7	7.0	3.0	5.1	1.8	Iris-versicolor
8	6.4	3.2	4.7	1.4	Iris-versicolor
9	6.9	3.1	5.0	1.5	Iris-versicolor
10	5.5	2.3	4.0	1.3	Iris-versicolor
11	6.5	2.8	5.2	1.9	Iris-versicolor
12	6.5	3.0	5.4	2.0	Iris-versicolor
13	6.5	3.2	5.1	1.8	Iris-versicolor
14	6.4	3.2	5.7	1.8	Iris-versicolor
15	6.0	3.0	4.9	1.5	Iris-virginica
16	5.9	3.0	4.8	1.4	Iris-virginica
17	6.1	3.0	5.9	2.1	Iris-virginica
18	6.3	2.5	5.8	1.8	Iris-virginica
19	6.3	2.9	5.6	1.9	Iris-virginica
20	6.4	3.2	6.5	2.0	Iris-virginica

Export code:

```
df.to_csv("path\new-name.csv")
```

Reading data from URL.

Define the col names

```
col_names = ["sepal-length-in-cm", "sepal-width-in-cm", "petal-length-in-cm", "petal-width-in-cm", "class"]
```

Read data from URL

```
iris_data = pd.read_csv(url, names=col_names)
```

iris_data.head()

21/03/24

5

Week 2

1. Performance Measures

2. Get the Data

2.1 Download the data.

```
import os
```

```
import tarfile
```

```
import urllib
```

```
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/master/"
```

```
Housing_path = os.path.join("data", "01")
```

```
Housing_URL = DOWNLOAD_ROOT + "dataset/housing/housing.tgz".
```

```
def fetch_housing_data(housing_url=Housing_URL,
                      housing_path=Housing_PATH):
```

```
os.makedirs(name=housing_path, exist_ok=True)
```

Download the data

```
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)
```

Create Test Set:

Train - 80%

Test - 20%

Using Numpy

```
import numpy as np
```

```
def split_train_test(data, test_ratio=0.2):  
    shuffled_indices = np.random.permutation(len(data))  
    train_indices = shuffled_indices[int(test_ratio * len(data)):]  
    test_indices = shuffled_indices[:int(test_ratio * len(data))]
```

Step 2: Discover and Visualize the Data
train-set.shape, test-set.shape

```
# Scatter plot  
housing.plot(kind='scatter', x='longitude', y='latitude')  
plt.show()
```

```
# Correlation  
housing[['population', 'median_house_value']].corr()
```

Step 3: Prepare the data for Machine Learning Algorithms.

```
housing = housing.drop(['median_house_value'], axis=1)  
housing = housing.drop(['median_income'], axis=1)
```

housing.shape; housing.dtypes

Data Cleaning

```
housing = housing.dropna(subset=['total_bedrooms'])
```

Step 4: Select and Train Model

from sklearn.linear_model

```
import LinearRegression  
lin_reg = LinearRegression()
```

from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing准备)

Better Evaluation

from sklearn.model_selection import cross_val_score

5

Step 5: Fine Tune Model

from sklearn.model_selection import

GridSearchCV

final_model = grid_search.best_estimator_

final_predictions = final_model.predict(
~~x=x_test_prepared~~)

Obj 11.1.2

(Modeling (contd.)

04/04/24

Date
Page

05

(Bivariate) linear regression = $y = mx + c$

(multiple) linear regression = $y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$

Implement linear and multiple linear Regression algorithm using appropriate dataset.

Linear Regression.

① df_val = pd.read_csv(r'c:/users/STUDENT/.csv')
df_val = head()

② plt.title('Salary Distribution Plot')
sns.distplot(df_val['Salary'])
plt.show()

③ plt.scatter(df_sal[['Years Experience']],
df_sal['Salary'], color='lightcoral')
plt.xlabel('Years of Exp')
plt.ylabel('Salary')
plt.box(False)
plt.show()

④ Splitting variables
~~x=df_sal.iloc[:, :-1]~~
~~y=df_sal.iloc[:, -1]~~

⑤ x-train, x-test, y-train, y-test = train-test-split
(x, y, test_size=0.2, random_state=0)

⑥ Regressor.get(x-train, y-train)

⑦ Prelim

5

$y\text{-pred-test} = \text{regressor.predict}(x\text{-test})$
 $y\text{-pred-train} = \text{regressor.predict}(x\text{-train})$

Steps.

- 1 Import libraries
- 2 Import data
- 3 Analyze data.
- 4 Split data
- 5 Predict results
- 6 Visualize predictions

Multiple Linear Regression

① $df\text{-start} = \text{pd.read_csv('E:/content/SD-start.csv')}$
 $df\text{-start.head()}$

② $plt.title('Profit Distribution Plot')$
 $sns.distplot(df\text{-start['Profit']})$
 $plt.show()$

③ ~~splitting~~
 $x = df\text{-start.iloc[:, :-1].values}$
 $y = df\text{-start.iloc[:, -1].values}$

④ $x\text{-train}, x\text{-test}, y\text{-train}, y\text{-test} = \text{Train-Test split}(x, y, \text{test_size}=0.2, \text{random_state}=0)$

⑤ $\text{Regressor} = \text{linear regression}()$
 $\text{regressor.fit}(x\text{-train}, y\text{-train})$

⑥ Predict

$y\text{-pred} = \text{regressor.predict}(x\text{-test})$

Steps:

1. Import libraries
2. Import data
3. Analyze data
4. Split into independent/dependent variables
5. Predict Results
6. Compare predictions.

Day 14/24

(Machine Learning) - Linear Regression

Importing Data (Head, tail, etc.)
Example: 27 (0.00000000000000027)

X = Standardized variable from training - 0.812

(Feature must be linearly related to target)

(Linear regression) prediction = Y

[Linear] output = Y

This type of function is called a linear function

(C) h(x) = mx + b (m = slope, b = intercept)

where m is the slope and b is the intercept

and h(x) is the predicted value for x

A curve (exponential, quadratic, logistic)

(Non-linear function) fit. with data - f(x)

• Non-linear function

(Non-linear function) non-linear fit. non-linear

(Non-linear function)

18/04/2024

Week - 4

```
import pandas as pd  
from sklearn.model_selection import train_test_split  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.tree import DecisionTreeClassifier,  
plot_tree  
from sklearn.metrics import accuracy_score  
import matplotlib.pyplot as plt
```

```
iris_data = pd.read_csv('Iris.csv')  
iris_data.head()  
iris_data.info()
```

<class 'pandas.core.frame.DataFrame'>

```
iris_data.drop('Id', inplace=True, axis=1)  
X = iris_data.drop(['Species'], axis=1)  
y = iris_data['Species']
```

X-train, X-test, y-train, y-test = train_test_split
(X, y, test_size=0.4, random_state=42)

~~dt_classifier = DecisionTreeClassifier(criterion='entropy', random_state=42)~~

~~dt_classifier.fit(X-train, y-train)~~

~~DecisionTreeClassifier~~
~~DecisionTreeClassifier(criterion='entropy', random_state=42)~~

plt.figure(figsize=(12, 8))

plot_tree(dt_classifier, feature_names=x.columns, class_names=dt_classifier.classes_, filled=True)

plt.show()

petalLengthCm <= 2.45 west

entropy = 1.581

samples 290

value = [27, 31, 32]

class = Iris-Virginica No

entropy = 0.0

petalWidthCm <= 0.95

samples 227 with entropy = 0

value = [27, 0, 0]

class = Iris-Setsosa

value = [0, 31, 32]

class = Iris-Virginica



y_pred = dt_classifier.predict(x-test)

accuracy = accuracy_score(y-test, y-pred)

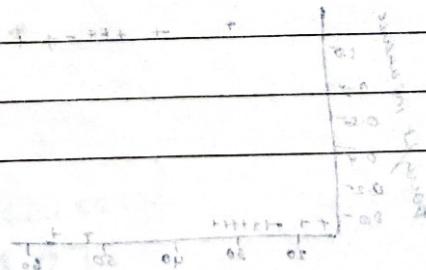
~~print(f'Accuracy: {accuracy}')~~

Accuracy: 0.9833

Chap14bun

(Comments aligned ex. SPA) 0.111 - 0.119

() 0.12 - 0.19



25/04/2024

(Q.1) [Set - 5] :-

```
import pandas as pd  
from matplotlib import pyplot as plt  
from sklearn.model_selection import train_test_  
split  
from sklearn.linear_model import Logistic  
Regression.
```

Read the data:

```
df = pd.read_csv("insurance_data.csv")
```

Display the first few rows of the DF
print(df.head(5))

	age	bought-insurance
0	22	0
1	25	0
2	32	1
3	38	0
4	46	1

Scatter plot

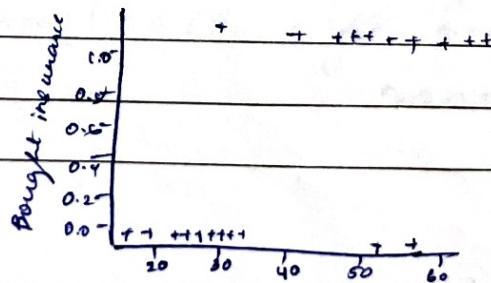
```
plt.scatter(df['age'], df['bought-insurance'],  
marker='+', color='red')
```

```
plt.xlabel('Age')
```

```
plt.ylabel('Bought Insurance')
```

```
plt.title('Age vs Bought Insurance')
```

```
plt.show()
```



`X-train, X-test, y-train, y-test = train_test_split(dt[['age']], dt['bought_insurance'], train_size=0.8)`

`print("X-test:")`
`print(X-test)`

`X-test:`

	age
14	49
24	50
16	25
2	47
6	55
22	40

`model = LogisticRegression()`

`model.fit(X-train, y-train)`

`print("\nX-test after fitting the model")`
`print(X-test)`

~~`y-predicted = model.predict(X-test)`~~
~~`print("\nPredicted values:")`~~
~~`print(y-predicted)`~~

`X-test after fitting the model:`

	age
14	49
24	50
16	25
2	47
6	55
22	40

`Predicted values:`

`[# 1 1 0 1 1 0]`

5 11 12 13 14 15 16 17 18 19 20

Probabilities

```
# Probabilities  
print('In Probabilities: ')  
print(model.predict_proba(x-test))
```

Model Accuracy (first 20%) 0.697
(all) 0.693

```
# Model Accuracy  
print ("Model Accuracy: " + str(accuracy))
```

```
print ("Model Accuracy : ", model.score(x-test, y-test))
```

Probabilities:

1 [0.21294835 0.78705165]

~~0.13694586 0.82305464~~

$$[0.98538887 \quad 0.01461113]$$

Model Accuracy:

0.5

25/4/24

(1) ~~debtors~~ ~~stripes~~ = ~~shark~~

~~(Montgomery, 1982)~~ fit 130m

$\beta_{1234} \text{ and part of } \alpha_{1234} \text{ in } \mathcal{L}(\text{triv})$

the (cost x) library is based on the library
of codes (mean for the first and last) taking
into account the (mean for the first and last)
and the (mean for the first and last).

obtainable with positive results.

With thanks to the Royal Society

2

88 9

26

20

卷之三

22

55

best dear

0113

09/06/2024

Date
Page 13

Week 6

KNN and SVM

```
import pandas as pd.
```

```
from matplotlib import pyplot as plt
```

```
import seaborn as sns
```

```
%matplotlib inline
```

Read the Dataset

```
df = pd.read_csv("content/Iris.csv")
```

Show the Plot:

```
plt.show()
```

```
plt.figure(figsize=(10, 6))
```

```
sns.Scatterplot(x=df["Petal Length"],  
y=df["Sepal Width"],  
legend=False, hue=df["Species"])
```

Add labels and Title to Plot:

```
plt.xlabel("Petal Length")
```

```
plt.ylabel("Sepal Width")
```

```
plt.title("Petal Length vs Sepal Width")
```

```
plt.show()
```

KNN:

```
from sklearn.model_selection import train-test-split as tts,
```

```
from sklearn.neighbors import KNeighborsClassifier
```

① X_train, X_test, Y_train, Y_test = tts(x, y, 0.6, 0)

neigh = KNeighborsClassifier(n_neighbors=2)

```
neigh.fit(X, y)
```

5
stud
ch. 10

page 10

$y_pred = \text{neigh.predict}(x\text{-test})$
 $\text{acc} = \text{accuracy_score}(y\text{-pred}, y\text{-test})$
 print acc
 $\rightarrow 1.0.$

from sklearn.svm import SVC

model = SVC()

model.fit(x, y)

$y_pred = \text{model.predict}(x\text{-test})$
 $\text{acc} = \text{accuracy_score}(y\text{-pred}, y\text{-test})$

print acc

1.0

Day 23/5/24

Explain steps to solve problem
1. Load data
2. Preprocess data
3. Train model
4. Evaluate model

Goal of ML has made it

(("Digit Recognition"))
("Digit Recognition")
("Digit Recognition")
("Digit Recognition")
("Digit Recognition")
("Digit Recognition")

- And defn. Recognise handwritten digits and
- Add noise

and digit recognition from handwritten digits

(e.g., 1, 2, 3, 4, 5, 6, 7, 8, 9) and P, Q, R, S, T, and X

(e.g., handwritten) original model predict a digit

(e.g., 4) if predict

23/05/2024

Week - 7Random Forest

pip install scikit-learn

import pandas as pd

from sklearn.model_selection import train-test
split from sklearn.metrics import accuracy_score

iris.load_iris()

x = iris.data

y = iris.target

15/265 KLP

x_train, x-test, y-train, y-test = train-test-split
(x, y, test_size=0.5, random_state=32)

rf_classifier = RandomForestClassifier()

rf_classifier.fit(x_train, y-train)

accuracy = accuracy_score(y-test, y-pred)

print("Accuracy: " accuracy)

Accuracy = 0.9333

classification_report = classification_report(y-test, y-pred)

print("\nClassification Report\n", classification_report)

Output

Output :

Classification Report

	Precision	Recall	f1-score	Support
0	1.00	1.00	1.00	30
1	0.81	1.00	0.89	21
2	1.00	0.79	0.88	24
Macro avg	0.94	0.93	0.93	75
WT avg	0.95	0.93	0.93	75

July 30/5/24

ANN using BP

import numpy as np

x = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)

y = np.array(([92], [85], [89]), dtype=float)

x = x / np.max(x, axis=0)

y = y / 100

epoch = 8000

lr = 0.1

inputlayer_neurons = 2

hiddenlayer_neurons = 3

output_neurons = 1

wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))

bh = np.random.uniform(size=(1, hiddenlayer_neurons))

wout = np.random.uniform(size=(hiddenlayer_neurons, 1))

bo = np.random.uniform(size=(1, output_neurons))

def sigmoid(x):

return 1 / (1 + np.exp(-x))

def derivatives_sigmoid(x):

return x * (1 - x)

for i in range(epoch):
 hinp1 = np.dot(x, wh)

5

$$\text{hinp} = \text{hinp} + b\text{h}^2$$

hlayer.act = sigmoid(hinp)

outinp = hinp.dot(hlayer.act, wout)

outinp = outinp + bout

output = sigmoid(outinp)

hiddengrad = derivatives.sigmoid(hlayer.act)

d-hiddenlayer = En * hiddengrad

print("Input: " + str(x))

print("Actual output: " + str(y))

print("Predicted output\n", output).

Output

Input

$\begin{bmatrix} 0.66667 \\ 0.3333 \\ 1 \end{bmatrix}$

Actual Output

$\begin{bmatrix} 0.92 \\ 0.86 \\ 0.89 \end{bmatrix}$

Predicted Output

$\begin{bmatrix} 0.78418718 \\ 0.72539013 \\ 0.7892728 \end{bmatrix}$

20/5/24

ADA -> Boost

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

$\text{iris} = \text{load_iris}()$ is a python function

$x = \text{iris}.\text{data}$

$y = \text{iris}.\text{target}$ \rightarrow target variable = 264

(train size) 50% and test = 50%

$X = \text{train}, X = \text{test}, y = \text{train}, y = \text{test}$ \rightarrow train-test-split
 $(x, y, size=0.4)$

$\text{adaboost_clf} = \text{AdaBoostClassifier}(n_estimators=30,$
 $\text{learning_rate}=1.0, \text{random_state}=42)$
 $\text{adaboost_clf} = \text{fit}(X=\text{train}, y=\text{train})$

$\text{accuracy} = \text{accuracy_score}(y=\text{test}, y=\text{pred})$

~~print("Accuracy = " + accuracy)~~

~~Output~~

~~Accuracy: 0.96666~~

~~ok~~

~~(s, s) daqdo - 11g~~

~~Accuracy: 0.96666~~

~~(s, s) daqdo - 11g~~

Week-8

```
import matplotlib.pyplot as plt  
from sklearn import datasets  
from sklearn.cluster import KMeans  
import pandas as pd  
import numpy as np  
  
iris = datasets.load_iris()  
x = pd.DataFrame(iris.data)  
x.columns = ['Sepal-length', 'Sepal-width', 'Petal-length',  
'Petal-width']  
y = pd.DataFrame(iris.target)  
y.columns = ['Targets']  
  
model = KMeans(n_clusters=3)  
model.fit(x)  
  
plt.figure(figsize=(14,14))  
colormap = np.array(['red', 'lime', 'black'])  
  
plt.subplot(2, 2, 1)  
plt.scatter(x['petal-length'], x['petal-width'],  
c=colormap[y['Targets']], s=40)  
plt.title('Real clusters')  
plt.xlabel('Petal length')  
plt.ylabel('Petal width')  
  
plt.subplot(2, 2, 2)  
plt.scatter(x['Petal-length'], x['Petal-width'],  
c=colormap[model.labels_], s=40)  
plt.title('K-means Clustering')
```

plt. x-label ('Petal length')

plt. y-label ('Petal width')

~~Scatter~~

Scatter plot showing relationship between Petal length and Petal width.

The scatter plot shows a strong positive linear correlation between Petal length and Petal width.

The data points are tightly clustered around the line of best fit.

The line of best fit passes through approximately (1.5, 0.1), (4.5, 1.5), and (5.5, 2.5).

The equation for the line of best fit is approximately $y = 0.3x + 0.1$.

Scatter plot showing relationship between Petal length and Sepal width.

The scatter plot shows a strong negative linear correlation between Petal length and Sepal width.

The data points are tightly clustered around the line of best fit.

The line of best fit passes through approximately (1.5, 0.1), (4.5, 1.5), and (5.5, 2.5).

The equation for the line of best fit is approximately $y = -0.3x + 0.1$.

([Sepal Length] vs [Petal Length])

([Sepal Width] vs [Petal Width])

([Sepal Length] vs [Sepal Width])

([Sepal Length] vs [Petal Length])

Scatter plot showing relationship between Sepal length and Sepal width.

The scatter plot shows a strong positive linear correlation between Sepal length and Sepal width.

The data points are tightly clustered around the line of best fit.

The line of best fit passes through approximately (1.5, 0.1), (4.5, 1.5), and (5.5, 2.5).

The equation for the line of best fit is approximately $y = 0.3x + 0.1$.

Scatter plot showing relationship between Sepal length and Petal length.

The scatter plot shows a strong positive linear correlation between Sepal length and Petal length.

Implement Dimensionality reduction using
PCA method:

```
import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np  
import seaborn as sns  
%matplotlib inline
```

```
from sklearn.datasets import load_breast_cancer  
cancer = load_breast_cancer()  
cancer.keys()
```

~~print(~~

OP:
`dict_keys(['DESCR', 'data', 'feature_names',
'target_names', 'target'])`

```
print(cancer['DESCR'])
```

```
df = pd.DataFrame(cancer['data'],  
columns=cancer['feature_names'])
```

~~df.head()~~

OP:

	mean radius	mean texture	mean perimeter	mean area	mean symmetry	mean ...	worst radius	worst area
0	17.98	10.38	122.80	1001.0	0.2419	28.38	17.33	
1	20.57	17.97	132.90	1326.0	0.1812	24.99	23.81	
2	19.69	21.25	130.00	1203.0	0.2069	23.57	28.53	
3	11.42	20.38	77.58	386.1	0.2877	14.91	26.80	
4	20.29	14.34	135.10	1297.0	0.1807	22.59	16.67	

PCA Visualization

plt.figure(figsize=(8, 6))

colors = ['r', 'g', 'b']

species_unique = df['species'].unique()

for target, color in zip(species_unique, colors):

indices_to_keep = pca_df['species'] == target

plt.scatter(pca_df.loc[indices_to_keep, 'Principal Component 1'],

pca_df.loc[indices_to_keep, 'Principal Component 2'],

c=color,

s=50)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title(title)

plt.legend(species_unique)

plt.grid()

plt.show()

explained_variance = pca.explained_variance_ratio_

print(f"Explained variance for title: {title} is {explained_variance[0]}")

Explained variance is 0.494

perform_pca_and_plot(df, ['sepal-length',
'sepal-width'], 'PCA of Sepal Dimensions')

perform_pca_and_plot(df, ['petal-length',
'petal-width'], 'PCA of Petal Dimensions')

PCA of Sepal Dimensions



Ans 30/5/24

PCA of Petal Dimensions

