Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Parth Sandeep Dabholkar of Computer Department, Semester VII with Roll No. 2103032 has completed a course of the necessary experiments in the subject Machine Learning under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2024 - 2025

Teacher In- Charge

Head of the Department

Date 15/10/24

Principal

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.	Select appropriate dataset and perform processing: i) imputation, ii) Anomaly detection iii) Standardization iv) Normalization v) Encoding	24	24/07	
2.	Implement Gradient Descent Algorithm		31/07	
3.	Implement simple Linear Regression using analytical and ML method.		07/08	
4.	Implement Logistic Regression without SKLearn		14/08	
5.	Implement Decision Tree algorithm		28/08	
6.	Implement SVM for non-linear classification		28/08	
7.	Implement ensemble methods to contribute to combine diff. models		04/09	
8.	Implement Principal Component Analysis.		11/09	
9.	Case study		18/09	
10.	Assignment - 1		12/07	
11.	Assignment -2		19/07	
12.	Assignment - 3		27/09	
13.	Assignment - 4		04/10	
14.	Assignment - 5		11/10	

16/10/24

ml-exp-1

October 18, 2024

[1]:
```
# prompt: read the dataset csv file named titanic.csv from session storage
# first print only 20 rows from the dataset

import pandas as pd

# Assuming the file is named 'titanic.csv'
df = pd.read_csv('titanic.csv')

# Print the first 20 rows
print(df.head(20))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
```

4	Allen, Mr. William Henry	male	35.0	0		
5	Moran, Mr. James	male	NaN	0		
6	McCarthy, Mr. Timothy J	male	54.0	0		
7	Palsson, Master. Gosta Leonard	male	2.0	3		
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0		
9	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1		
10	Sandstrom, Miss. Marguerite Rut	female	4.0	1		
11	Bonnell, Miss. Elizabeth	female	58.0	0		
12	Saundercock, Mr. William Henry	male	20.0	0		
13	Andersson, Mr. Anders Johan	male	39.0	1		
14	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0		
15	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0		
16	Rice, Master. Eugene	male	2.0	4		
17	Williams, Mr. Charles Eugene	male	NaN	0		
18	Vander Planke, Mrs. Julius (Emelia Maria Vande…	female	31.0	1		
19	Masselmani, Mrs. Fatima	female	NaN	0		

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S
5	0	330877	8.4583	NaN	Q
6	0	17463	51.8625	E46	S
7	1	349909	21.0750	NaN	S
8	2	347742	11.1333	NaN	S
9	0	237736	30.0708	NaN	C
10	1	PP 9549	16.7000	G6	S
11	0	113783	26.5500	C103	S
12	0	A/5. 2151	8.0500	NaN	S
13	5	347082	31.2750	NaN	S
14	0	350406	7.8542	NaN	S
15	0	248706	16.0000	NaN	S
16	1	382652	29.1250	NaN	Q
17	0	244373	13.0000	NaN	S
18	0	345763	18.0000	NaN	S
19	0	2649	7.2250	NaN	C

PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0

```
Cabin          687
Embarked         2
dtype: int64
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```

[2]:
```
# prompt: perform only one technique of imputation on the above dataset

# Fill missing values in the 'Age' column with the mean age
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Print the first 20 rows to see the imputed values
print(df.head(20))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3

                                                 Name     Sex         Age  \
```

```
0                            Braund, Mr. Owen Harris    male  22.000000
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.000000
2                             Heikkinen, Miss. Laina  female  26.000000
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.000000
4                           Allen, Mr. William Henry    male  35.000000
5                                   Moran, Mr. James    male  29.699118
6                            McCarthy, Mr. Timothy J    male  54.000000
7                     Palsson, Master. Gosta Leonard    male   2.000000
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.000000
9                 Nasser, Mrs. Nicholas (Adele Achem)  female  14.000000
10                Sandstrom, Miss. Marguerite Rut  female   4.000000
11                       Bonnell, Miss. Elizabeth  female  58.000000
12                 Saundercock, Mr. William Henry    male  20.000000
13                    Andersson, Mr. Anders Johan    male  39.000000
14            Vestrom, Miss. Hulda Amanda Adolfina  female  14.000000
15                Hewlett, Mrs. (Mary D Kingcome)   female  55.000000
16                            Rice, Master. Eugene    male   2.000000
17                    Williams, Mr. Charles Eugene    male  29.699118
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…  female  31.000000
19                           Masselmani, Mrs. Fatima  female  29.699118

    SibSp  Parch            Ticket     Fare Cabin Embarked
0       1      0         A/5 21171   7.2500   NaN        S
1       1      0          PC 17599  71.2833   C85        C
2       0      0  STON/O2. 3101282   7.9250   NaN        S
3       1      0            113803  53.1000  C123        S
4       0      0            373450   8.0500   NaN        S
5       0      0            330877   8.4583   NaN        Q
6       0      0             17463  51.8625   E46        S
7       3      1            349909  21.0750   NaN        S
8       0      2            347742  11.1333   NaN        S
9       1      0            237736  30.0708   NaN        C
10      1      1          PP 9549  16.7000    G6        S
11      0      0            113783  26.5500  C103        S
12      0      0         A/5. 2151   8.0500   NaN        S
13      1      5            347082  31.2750   NaN        S
14      0      0            350406   7.8542   NaN        S
15      0      0            248706  16.0000   NaN        S
16      4      1            382652  29.1250   NaN        Q
17      0      0            244373  13.0000   NaN        S
18      1      0            345763  18.0000   NaN        S
19      0      0              2649   7.2250   NaN        C
```

[7]:
```
# prompt: now perform KNN imputation technique

import pandas as pd
from sklearn.impute import KNNImputer
```

```
# Create a KNN imputer object
imputer = KNNImputer(n_neighbors=5)

# Select the columns with missing values
df_numeric = df[['Age', 'Fare']]

# Fit and transform the data
df_imputed = imputer.fit_transform(df_numeric)

# Convert the imputed data back to a DataFrame
df_imputed = pd.DataFrame(df_imputed, columns=['Age', 'Fare'])

# Replace the original columns with the imputed values
df[['Age', 'Fare']] = df_imputed[['Age', 'Fare']]

# Print the first 20 rows to see the imputed values
print(df.head(20))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3

                                                 Name     Sex        Age  \
0                             Braund, Mr. Owen Harris    male  22.000000
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.000000
2                              Heikkinen, Miss. Laina  female  26.000000
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.000000
4                            Allen, Mr. William Henry    male  35.000000
```

```
5                                    Moran, Mr. James    male  29.699118
6                             McCarthy, Mr. Timothy J    male  54.000000
7                      Palsson, Master. Gosta Leonard    male   2.000000
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.000000
9                  Nasser, Mrs. Nicholas (Adele Achem)  female  14.000000
10                     Sandstrom, Miss. Marguerite Rut  female   4.000000
11                            Bonnell, Miss. Elizabeth  female  58.000000
12                     Saundercock, Mr. William Henry    male  20.000000
13                        Andersson, Mr. Anders Johan    male  39.000000
14                Vestrom, Miss. Hulda Amanda Adolfina  female  14.000000
15                    Hewlett, Mrs. (Mary D Kingcome)   female  55.000000
16                                Rice, Master. Eugene    male   2.000000
17                        Williams, Mr. Charles Eugene    male  29.699118
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…  female  31.000000
19                             Masselmani, Mrs. Fatima  female  29.699118

    SibSp  Parch            Ticket     Fare Cabin Embarked
0       1      0         A/5 21171   7.2500   NaN        S
1       1      0          PC 17599  71.2833   C85        C
2       0      0  STON/O2. 3101282   7.9250   NaN        S
3       1      0            113803  53.1000  C123        S
4       0      0            373450   8.0500   NaN        S
5       0      0            330877   8.4583   NaN        Q
6       0      0             17463  51.8625   E46        S
7       3      1            349909  21.0750   NaN        S
8       0      2            347742  11.1333   NaN        S
9       1      0            237736  30.0708   NaN        C
10      1      1          PP 9549  16.7000    G6        S
11      0      0            113783  26.5500  C103        S
12      0      0         A/5. 2151   8.0500   NaN        S
13      1      5            347082  31.2750   NaN        S
14      0      0            350406   7.8542   NaN        S
15      0      0            248706  16.0000   NaN        S
16      4      1            382652  29.1250   NaN        Q
17      0      0            244373  13.0000   NaN        S
18      1      0            345763  18.0000   NaN        S
19      0      0              2649   7.2250   NaN        C
```

[10]:
```python
# prompt: perform Anomaly Detection on the above dataset
# and give which column and row was found to be an anamoly till first 20 rows

import pandas as pd
from sklearn.ensemble import IsolationForest

# Assuming the file is named 'titanic.csv'
df = pd.read_csv('titanic.csv')
```

```
# Fill missing values in the 'Age' column with the mean age
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Select relevant numerical features for anomaly detection
features = ['Age', 'Fare']
df_selected = df[features]

# Create an Isolation Forest model
model = IsolationForest()

# Fit the model to the data
model.fit(df_selected)

# Predict anomalies
predictions = model.predict(df_selected)

# Get the indices of anomalies
anomaly_indices = [i for i, pred in enumerate(predictions) if pred == -1]

# Print the indices of anomalies within the first 20 rows
for index in anomaly_indices:
  if index < 20:
    print(f"Anomaly detected at row {index} in columns: {features}")
```

```
Anomaly detected at row 6 in columns: ['Age', 'Fare']
Anomaly detected at row 7 in columns: ['Age', 'Fare']
Anomaly detected at row 10 in columns: ['Age', 'Fare']
Anomaly detected at row 11 in columns: ['Age', 'Fare']
Anomaly detected at row 16 in columns: ['Age', 'Fare']
```

[11]:
```
# prompt: perform Standardization on the above dataset

import pandas as pd
from sklearn.preprocessing import StandardScaler

# Select numerical features for standardization
numerical_features = ['Age', 'Fare']
df_numerical = df[numerical_features]

# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler to the data and transform it
df_standardized = scaler.fit_transform(df_numerical)

# Convert the standardized data back to a DataFrame
df_standardized = pd.DataFrame(df_standardized, columns=numerical_features)
```

```
# Replace the original numerical columns with the standardized values
df[numerical_features] = df_standardized

# Print the first 20 rows to see the standardized values
print(df.head(20))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3

                                                 Name     Sex       Age  \
0                             Braund, Mr. Owen Harris    male -0.592481
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  0.638789
2                              Heikkinen, Miss. Laina  female -0.284663
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  0.407926
4                            Allen, Mr. William Henry    male  0.407926
5                                    Moran, Mr. James    male  0.000000
6                             McCarthy, Mr. Timothy J    male  1.870059
7                      Palsson, Master. Gosta Leonard    male -2.131568
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female -0.207709
9                 Nasser, Mrs. Nicholas (Adele Achem)  female -1.208115
10                    Sandstrom, Miss. Marguerite Rut  female -1.977659
11                           Bonnell, Miss. Elizabeth  female  2.177876
12                     Saundercock, Mr. William Henry    male -0.746389
13                        Andersson, Mr. Anders Johan    male  0.715743
14               Vestrom, Miss. Hulda Amanda Adolfina  female -1.208115
15                   Hewlett, Mrs. (Mary D Kingcome)   female  1.947013
16                               Rice, Master. Eugene    male -2.131568
```

```
17                      Williams, Mr. Charles Eugene    male  0.000000
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…  female  0.100109
19                           Masselmani, Mrs. Fatima  female  0.000000

    SibSp  Parch            Ticket      Fare Cabin Embarked
0       1      0         A/5 21171 -0.502445   NaN        S
1       1      0          PC 17599  0.786845   C85        C
2       0      0  STON/O2. 3101282 -0.488854   NaN        S
3       1      0            113803  0.420730  C123        S
4       0      0            373450 -0.486337   NaN        S
5       0      0            330877 -0.478116   NaN        Q
6       0      0             17463  0.395814   E46        S
7       3      1            349909 -0.224083   NaN        S
8       0      2            347742 -0.424256   NaN        S
9       1      0            237736 -0.042956   NaN        C
10      1      1          PP 9549 -0.312172    G6        S
11      0      0            113783 -0.113846  C103        S
12      0      0         A/5. 2151 -0.486337   NaN        S
13      1      5            347082 -0.018709   NaN        S
14      0      0            350406 -0.490280   NaN        S
15      0      0            248706 -0.326267   NaN        S
16      4      1            382652 -0.061999   NaN        Q
17      0      0            244373 -0.386671   NaN        S
18      1      0            345763 -0.285997   NaN        S
19      0      0              2649 -0.502949   NaN        C
```

[12]:
```python
# prompt: perform normalization on the above dataset

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Select numerical features for normalization
numerical_features = ['Age', 'Fare']
df_numerical = df[numerical_features]

# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Fit the scaler to the data and transform it
df_normalized = scaler.fit_transform(df_numerical)

# Convert the normalized data back to a DataFrame
df_normalized = pd.DataFrame(df_normalized, columns=numerical_features)

# Replace the original numerical columns with the normalized values
df[numerical_features] = df_normalized
```

```
# Print the first 20 rows to see the normalized values
print(df.head(20))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3

                                                 Name     Sex       Age  \
0                             Braund, Mr. Owen Harris    male  0.271174
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  0.472229
2                              Heikkinen, Miss. Laina  female  0.321438
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  0.434531
4                            Allen, Mr. William Henry    male  0.434531
5                                    Moran, Mr. James    male  0.367921
6                             McCarthy, Mr. Timothy J    male  0.673285
7                      Palsson, Master. Gosta Leonard    male  0.019854
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  0.334004
9                 Nasser, Mrs. Nicholas (Adele Achem)  female  0.170646
10                    Sandstrom, Miss. Marguerite Rut  female  0.044986
11                           Bonnell, Miss. Elizabeth  female  0.723549
12                     Saundercock, Mr. William Henry    male  0.246042
13                        Andersson, Mr. Anders Johan    male  0.484795
14               Vestrom, Miss. Hulda Amanda Adolfina  female  0.170646
15                   Hewlett, Mrs. (Mary D Kingcome)   female  0.685851
16                               Rice, Master. Eugene    male  0.019854
17                       Williams, Mr. Charles Eugene    male  0.367921
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…  female  0.384267
19                            Masselmani, Mrs. Fatima  female  0.367921
```

```
    SibSp  Parch            Ticket      Fare Cabin Embarked
0       1      0         A/5 21171  0.014151   NaN        S
1       1      0          PC 17599  0.139136   C85        C
2       0      0  STON/O2. 3101282  0.015469   NaN        S
3       1      0            113803  0.103644  C123        S
4       0      0            373450  0.015713   NaN        S
5       0      0            330877  0.016510   NaN        Q
6       0      0             17463  0.101229   E46        S
7       3      1            349909  0.041136   NaN        S
8       0      2            347742  0.021731   NaN        S
9       1      0            237736  0.058694   NaN        C
10      1      1          PP 9549  0.032596    G6        S
11      0      0            113783  0.051822  C103        S
12      0      0         A/5. 2151  0.015713   NaN        S
13      1      5            347082  0.061045   NaN        S
14      0      0            350406  0.015330   NaN        S
15      0      0            248706  0.031230   NaN        S
16      4      1            382652  0.056848   NaN        Q
17      0      0            244373  0.025374   NaN        S
18      1      0            345763  0.035134   NaN        S
19      0      0              2649  0.014102   NaN        C
```

[13]:
```
# prompt: perform Encoding on the above dataset

import pandas as pd
# One-hot encode the 'Sex' and 'Embarked' columns
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)

# Print the first 20 rows to see the encoded values
print(df.head(20))
```

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
```

```
16          17         0       3
17          18         1       2
18          19         0       3
19          20         1       3

                                                  Name       Age  SibSp  Parch  \
0                              Braund, Mr. Owen Harris  0.271174      1      0
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  0.472229      1      0
2                               Heikkinen, Miss. Laina  0.321438      0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  0.434531      1      0
4                             Allen, Mr. William Henry  0.434531      0      0
5                                     Moran, Mr. James  0.367921      0      0
6                              McCarthy, Mr. Timothy J  0.673285      0      0
7                    Palsson, Master. Gosta Leonard  0.019854      3      1
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  0.334004      0      2
9                 Nasser, Mrs. Nicholas (Adele Achem)  0.170646      1      0
10                    Sandstrom, Miss. Marguerite Rut  0.044986      1      1
11                           Bonnell, Miss. Elizabeth  0.723549      0      0
12                     Saundercock, Mr. William Henry  0.246042      0      0
13                        Andersson, Mr. Anders Johan  0.484795      1      5
14               Vestrom, Miss. Hulda Amanda Adolfina  0.170646      0      0
15                   Hewlett, Mrs. (Mary D Kingcome)   0.685851      0      0
16                               Rice, Master. Eugene  0.019854      4      1
17                       Williams, Mr. Charles Eugene  0.367921      0      0
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…  0.384267      1      0
19                             Masselmani, Mrs. Fatima  0.367921      0      0

              Ticket      Fare Cabin  Sex_male  Embarked_Q  Embarked_S
0          A/5 21171  0.014151   NaN      True       False        True
1           PC 17599  0.139136   C85     False       False       False
2   STON/O2. 3101282  0.015469   NaN     False       False        True
3             113803  0.103644  C123     False       False        True
4             373450  0.015713   NaN      True       False        True
5             330877  0.016510   NaN      True        True       False
6              17463  0.101229   E46      True       False        True
7             349909  0.041136   NaN      True       False        True
8             347742  0.021731   NaN     False       False        True
9             237736  0.058694   NaN     False       False       False
10           PP 9549  0.032596    G6     False       False        True
11            113783  0.051822  C103     False       False        True
12         A/5. 2151  0.015713   NaN      True       False        True
13            347082  0.061045   NaN      True       False        True
14            350406  0.015330   NaN     False       False        True
15            248706  0.031230   NaN     False       False        True
16            382652  0.056848   NaN      True        True       False
17            244373  0.025374   NaN      True       False        True
18            345763  0.035134   NaN     False       False        True
19              2649  0.014102   NaN     False       False       False
```

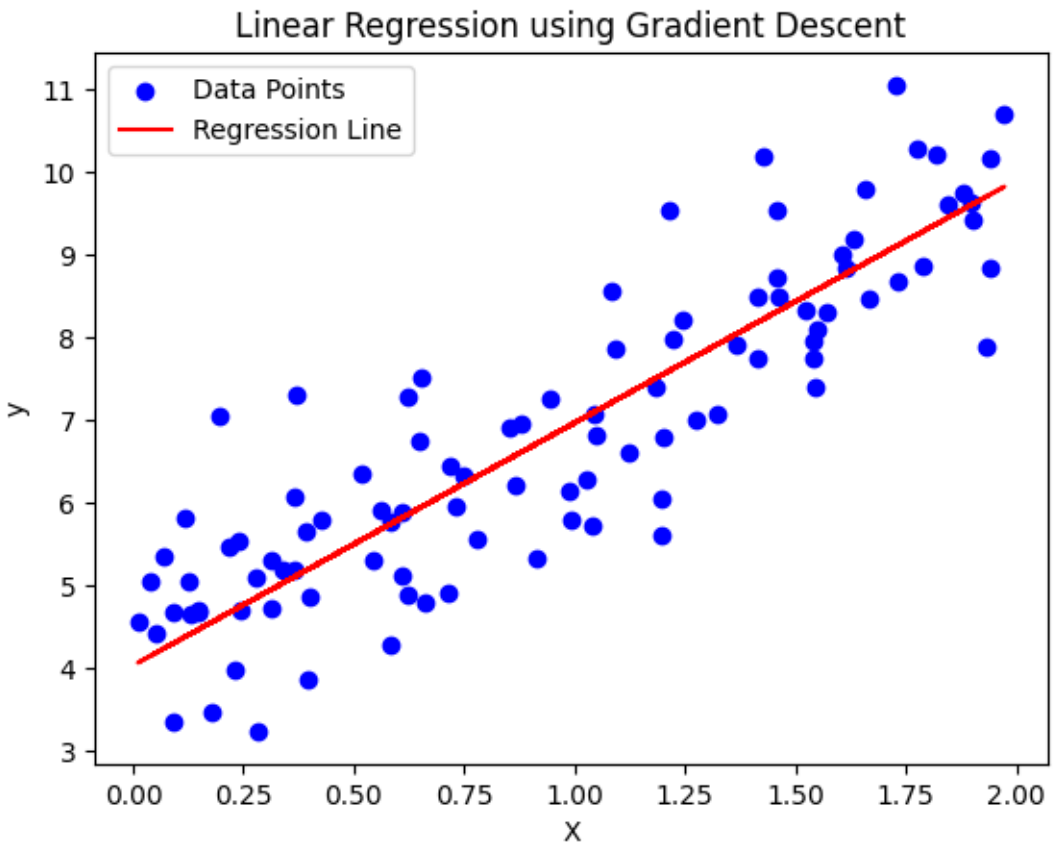
ml-exp-2

October 18, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

# Generate some data (or you can replace this with your own dataset)
# Example: y = 2x + 1 (with some noise)
np.random.seed(42)
X = 2 * np.random.rand(100, 1)  # Features
y = 4 + 3 * X + np.random.randn(100, 1)  # Target variable with noise

# Gradient Descent Parameters
learning_rate = 0.01
iterations = 1000
m = len(y)

# Initialize weights (slope) and bias (intercept)
theta_0 = 0  # bias
theta_1 = 0  # slope

# Gradient Descent function
def gradient_descent(X, y, theta_0, theta_1, learning_rate, iterations):
    m = len(y)
    for i in range(iterations):
        # Predicted value y_hat (hypothesis)
        y_hat = theta_0 + theta_1 * X

        # Cost function (Mean Squared Error)
        cost = (1 / (2 * m)) * np.sum((y_hat - y) ** 2)

        # Gradient calculation
        d_theta_0 = (1 / m) * np.sum(y_hat - y)
        d_theta_1 = (1 / m) * np.sum((y_hat - y) * X)

        # Update parameters
        theta_0 = theta_0 - learning_rate * d_theta_0
        theta_1 = theta_1 - learning_rate * d_theta_1

        if i % 100 == 0:  # Print cost every 100 iterations for reference
```

```
            print(f"Iteration {i}: Cost = {cost}")

    return theta_0, theta_1

# Run gradient descent
theta_0, theta_1 = gradient_descent(X, y, theta_0, theta_1, learning_rate,␣
 ↪iterations)
print(f"\nOptimized parameters: theta_0 = {theta_0}, theta_1 = {theta_1}")

# Predict using optimized parameters
y_pred = theta_0 + theta_1 * X

# RMSE (Root Mean Squared Error)
rmse = np.sqrt(np.mean((y_pred - y) ** 2))
print(f"\nRMSE = {rmse}")

# Plot the data and the regression line
plt.scatter(X, y, color='blue', label='Data Points')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.title('Linear Regression using Gradient Descent')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
Iteration 0: Cost = 25.00415182181856
Iteration 100: Cost = 0.8861489212467981
Iteration 200: Cost = 0.48475176090649796
Iteration 300: Cost = 0.45724927014472444
Iteration 400: Cost = 0.44164609589842474
Iteration 500: Cost = 0.43059828828478713
Iteration 600: Cost = 0.42273346673936874
Iteration 700: Cost = 0.41713392331624627
Iteration 800: Cost = 0.41314718744676066
Iteration 900: Cost = 0.4103087303503442

Optimized parameters: theta_0 = 4.033214213676792, theta_1 = 2.9307052510870104

RMSE = 0.9036457483647183
```

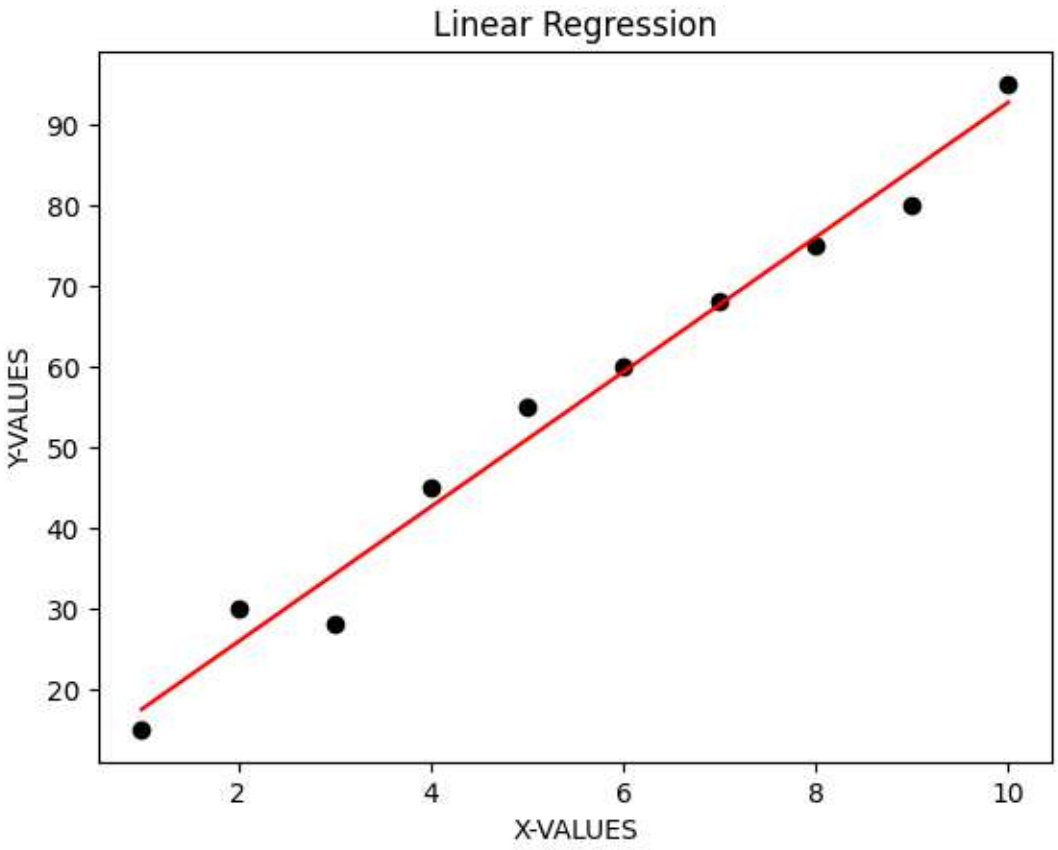


[2]:
```python
import matplotlib.pyplot as plt

def ml_caluclate_parameters(X, Y):
    mean_x = sum(X)/len(X)
    mean_y = sum(Y)/len(Y)

    numer = 0
    denom = 0

    for i in range(len(X)):
        numer = numer + (X[i] - mean_x)*(Y[i] - mean_y)
        denom = denom + (X[i] - mean_x)**2

    beta_1 = numer/denom

    beta_0 = mean_y - beta_1*mean_x

    return [beta_0, beta_1]
```

```
def predicted_values_usingML(X, Y):
    beta_0 = ml_caluclate_parameters(X, Y)[0]
    beta_1 = ml_caluclate_parameters(X, Y)[1]

    y_pred = []

    for i in range(len(X)):
        y_pred.append(round(beta_0 + beta_1*X[i], 4))

    return y_pred

def calculate_r_square(X, Y):
    y_pred = predicted_values_usingML(X, Y)
    y_mean = sum(Y)/len(Y)

    sum_y_total = 0
    sum_y_residual = 0
    for i in range(len(Y)):
        sum_y_total = sum_y_total + (Y[i] - y_pred[i])**2
        sum_y_residual = sum_y_residual + (Y[i] - y_mean)**2

    r_square = 1 - (sum_y_total / sum_y_residual)

    return r_square

    return mse

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Y = [15, 30, 28, 45, 55, 60, 68, 75, 80, 95]

y_pred = predicted_values_usingML(X, Y)
[beta_0, beta_1] = ml_caluclate_parameters(X, Y)
print(y_pred)
print(f"Calculated beta_0: {round(beta_0, 4)}\nCalculated beta_1:␣
 ↪{round(beta_1, 4)}")
print(f"R Squared: {calculate_r_square(X, Y)}")

plt.title("Linear Regression")

plt.xlabel("X-VALUES")
plt.ylabel("Y-VALUES")

plt.scatter(X, Y, color = "k")
plt.plot(X, y_pred, color = "r")
plt.show()
```

```
[17.4909, 25.8485, 34.2061, 42.5636, 50.9212, 59.2788, 67.6364, 75.9939,
```

```
84.3515, 92.7091]
Calculated beta_0: 9.1333
Calculated beta_1: 8.3576
R Squared: 0.9812099961960871
```



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

```
data = pd.read_csv('p_iris.csv')
```

```
x = data['SepalLengthCm'].values
y = data['SepalWidthCm'].values
```

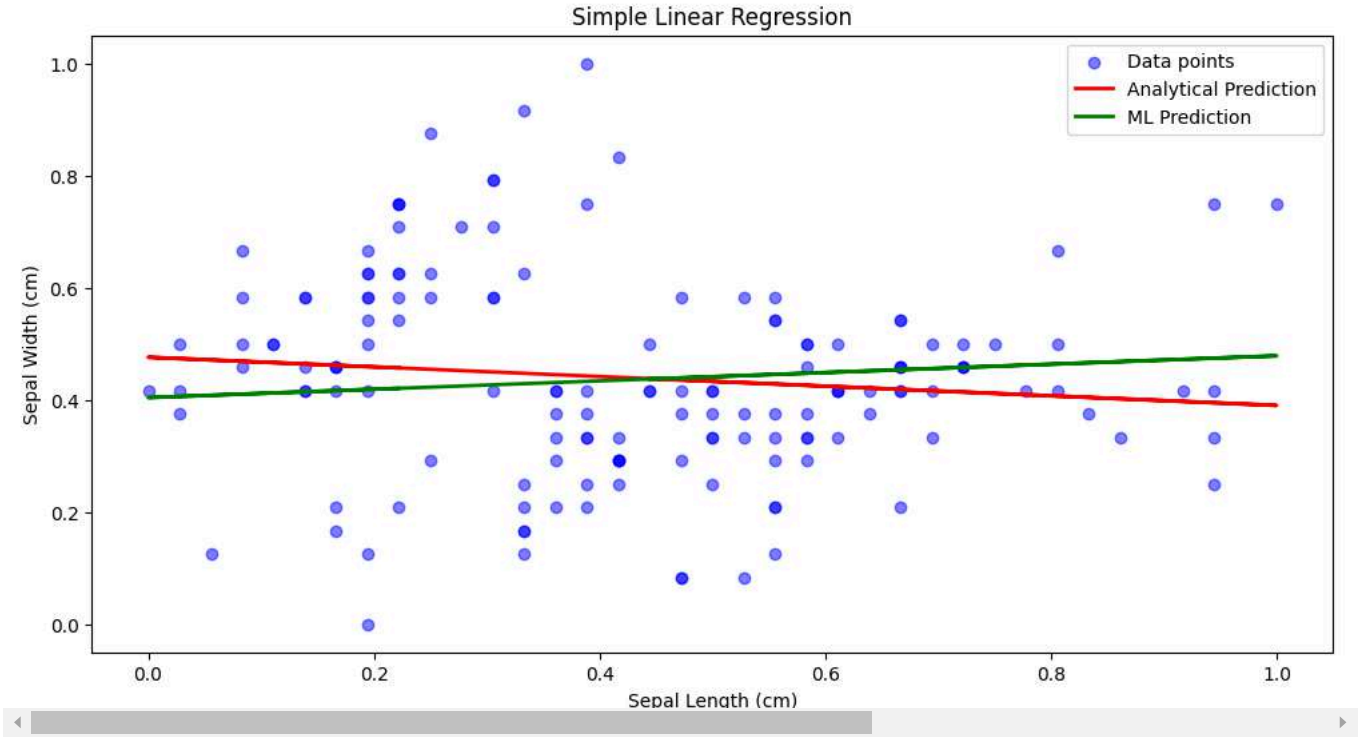
```
# 1. Analytical Method
# Calculate means
N = len(x)
mean_x = np.mean(x)
mean_y = np.mean(y)

# Calculate coefficients
beta_1 = (N * np.sum(x * y) - np.sum(x) * np.sum(y)) / (N * np.sum(x ** 2) - np.sum(x) ** 2)
beta_0 = mean_y - beta_1 * mean_x

print(f"Analytical Method: Intercept (β0) = {beta_0}, Slope (β1) = {beta_1}")

# Make predictions using the derived coefficients
y_pred_analytical = beta_0 + beta_1 * x
```

```
Analytical Method: Intercept (β0) = 0.47599332256678795, Slope (β1) = -0.08590235069574728
```

```
# 2. Machine Learning Method (Gradient Descent)
# Initialize parameters
alpha = 0.01  # Learning rate
iterations = 1000
m = len(x)  # Number of observations

# Initialize weights
theta_0 = 0  # Intercept
theta_1 = 0  # Slope

# Gradient Descent
for _ in range(iterations):
    # Predictions
    y_pred_ml = theta_0 + theta_1 * x
    # Calculate gradients
    error = y_pred_ml - y
    gradient_0 = (1/m) * np.sum(error)
    gradient_1 = (1/m) * np.sum(error * x)

    # Update weights
    theta_0 -= alpha * gradient_0
    theta_1 -= alpha * gradient_1

print(f"Machine Learning Method: Intercept (θ0) = {theta_0}, Slope (θ1) = {theta_1}")

# Make predictions using the ML method
y_pred_ml = theta_0 + theta_1 * x
```

```
Machine Learning Method: Intercept (θ0) = 0.40406666573537464, Slope (θ1) = 0.07447297472687159
```

```
# Plotting the results
plt.figure(figsize=(12, 6))
plt.scatter(x, y, color='blue', label='Data points', alpha=0.5)
plt.plot(x, y_pred_analytical, color='red', label='Analytical Prediction', linewidth=2)
plt.plot(x, y_pred_ml, color='green', label='ML Prediction', linewidth=2)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Simple Linear Regression')
plt.legend()
plt.show()
```


Simple Linear Regression
Data points
Analytical Prediction
ML Prediction
Sepal Width (cm)
Sepal Length (cm)
0.0
0.2
0.4
0.6
0.8
1.0

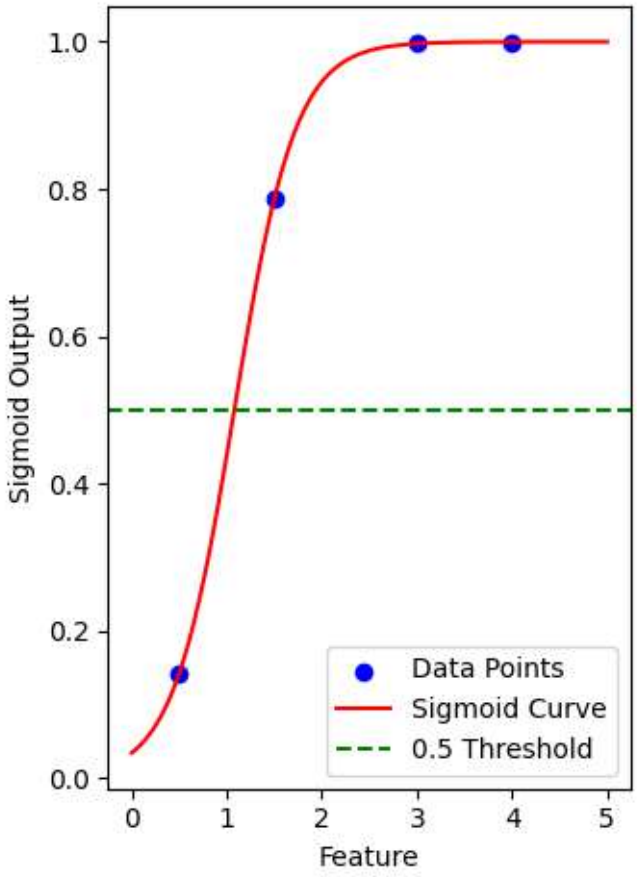
ml-exp-3-1

October 18, 2024

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt


def sigmoid(z):
    return 1 / (1 + np.exp(-z))


def logistic_regression(X, y, lr, iterations):
    m, n = X.shape
    weights = np.zeros(n)
    bias = 0

    for _ in range(iterations):

        z = np.dot(X, weights) + bias

        predictions = sigmoid(z)


        dw = (1/m) * np.dot(X.T, (predictions - y))
        db = (1/m) * np.sum(predictions - y)


        weights -= lr * dw
        bias -= lr * db

    return weights, bias


def predict(X, weights, bias):
    z = np.dot(X, weights) + bias
    return sigmoid(z) >= 0.5
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

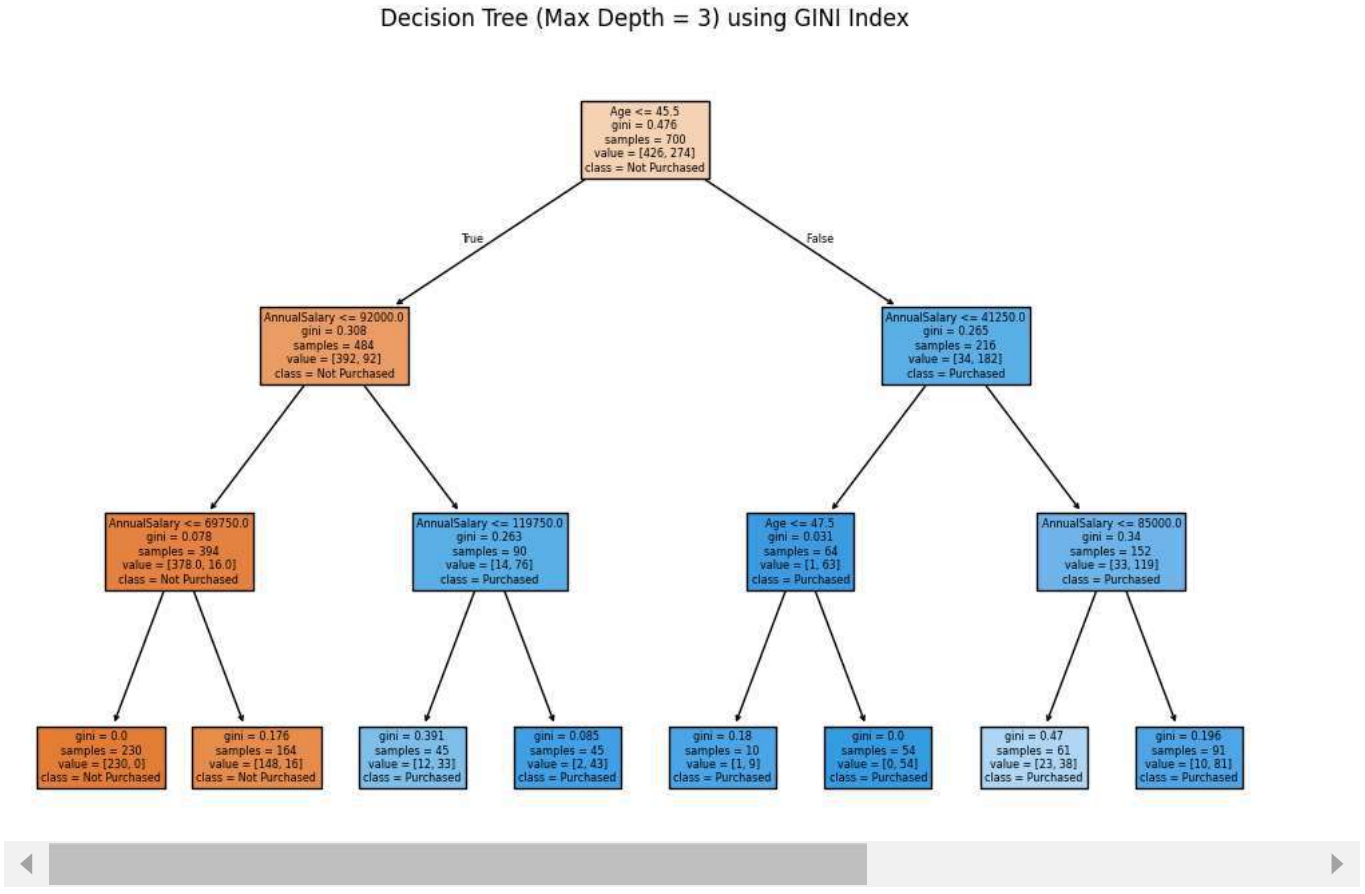
```
X = np.array([[0.5, 1], [1.5, 1], [3, 1], [4, 1]])
y = np.array([0, 0, 1, 1])




plt.subplot(1, 2, 2)
z_values = np.dot(X[:, 0], weights[0]) + bias
sigmoid = 1 / (1 + np.exp(-z_values))
plt.scatter(X[:, 0], sigmoid, color='blue', label='Data Points')
plt.plot(x_values, 1 / (1 + np.exp(-(weights[0] * x_values + bias))),␣
 ↪color='red', label='Sigmoid Curve')
plt.axhline(y=0.5, color='green', linestyle='--', label='0.5 Threshold')
plt.xlabel('Feature')
plt.ylabel('Sigmoid Output')
plt.legend()

plt.tight_layout()
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
Sigmoid Output
0
1
2
3
4
5
Feature
Data Points
Sigmoid Curve
0.5 Threshold

[ ]:

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

# Load the dataset
file_path = 'sample_data/car_data.csv'  # Update the file path if necessary
car_data = pd.read_csv(file_path)

# Preprocessing: Encode the 'Gender' column using LabelEncoder (Male=1, Female=0)
label_encoder = LabelEncoder()
car_data['Gender'] = label_encoder.fit_transform(car_data['Gender'])

# Splitting the data into features (X) and target (y)
X = car_data[['Gender', 'Age', 'AnnualSalary']]  # Features
y = car_data['Purchased']  # Target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Build the Decision Tree classifier with GINI index and max depth of 3
classifier_limited_depth = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
classifier_limited_depth.fit(X_train, y_train)

# Predict on the test set
y_pred = classifier_limited_depth.predict(X_test)

# Evaluate the model's accuracy
accuracy_limited_depth = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Decision Tree model (Max Depth = 3): {accuracy_limited_depth * 100:.2f}%")

# Plot the Decision Tree with max depth of 3
plt.figure(figsize=(12, 8))
tree.plot_tree(classifier_limited_depth, feature_names=['Gender', 'Age', 'AnnualSalary'], class_names=['Not Purchased', 'Purchased'], fil
plt.title("Decision Tree (Max Depth = 3) using GINI Index")
plt.show()
```

```
Accuracy of the Decision Tree model (Max Depth = 3): 89.33%
```



ml-exp-6

October 18, 2024

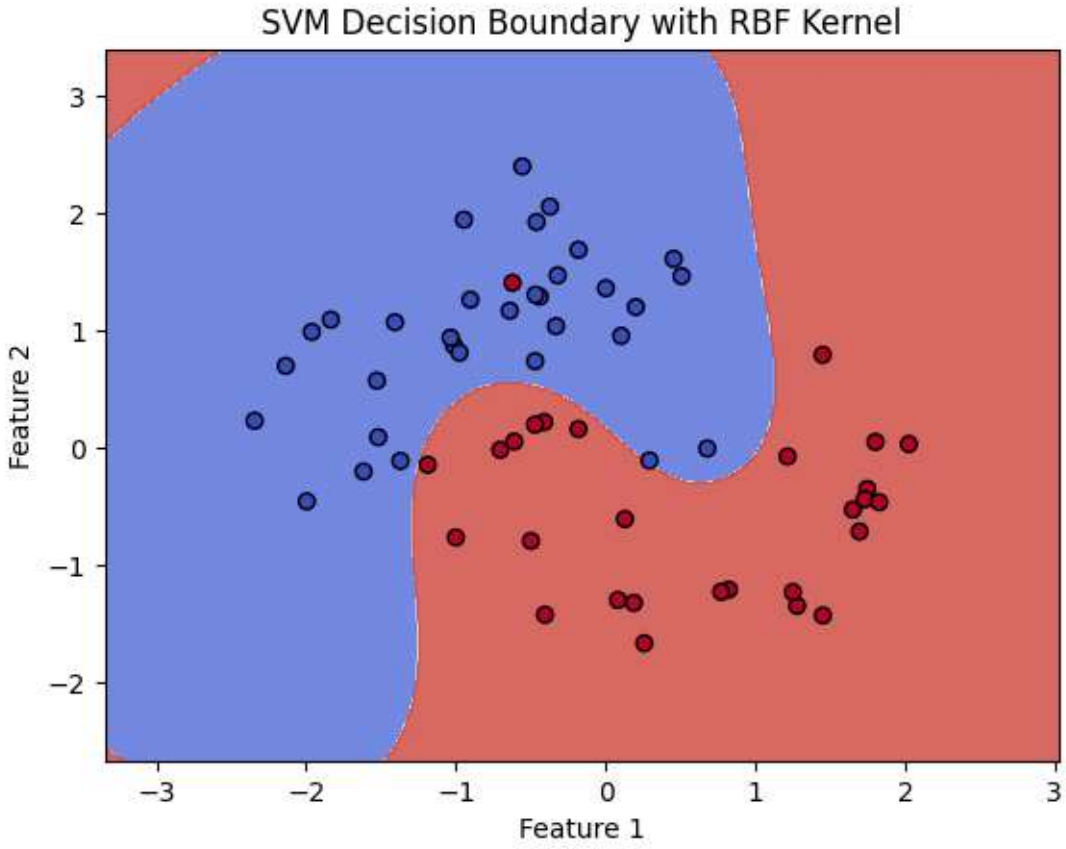
[ ]:
```
!pip install scikit-learn matplotlib
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.3.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: numpy<2.0,>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

[ ]:
```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load dataset
X, y = datasets.make_moons(n_samples=200, noise=0.2, random_state=42)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create an SVM classifier with RBF kernel
svm_classifier = SVC(kernel='rbf', gamma='scale', C=1.0)

# Train the classifier
svm_classifier.fit(X_train, y_train)

# Make predictions
y_pred = svm_classifier.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Function to visualize decision boundaries
def plot_decision_boundary(clf, X, y):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max,␣
 ↪0.01))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', cmap=plt.cm.coolwarm)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary with RBF Kernel')
    plt.show()

# Visualize the decision boundary
plot_decision_boundary(svm_classifier, X_test, y_test)
```

```
Accuracy: 98.33%
```

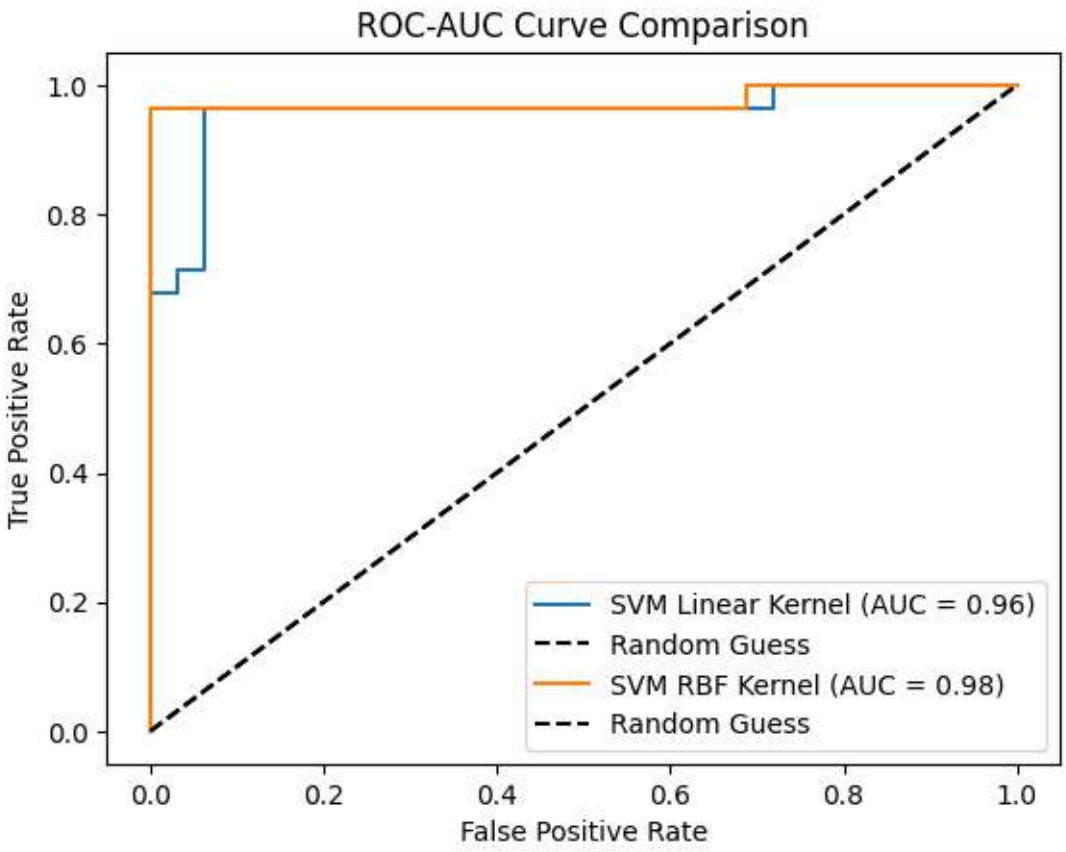


[ ]:
```python
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, roc_curve,␣
↪auc, confusion_matrix

# Load dataset (e.g., moon dataset)
X, y = datasets.make_moons(n_samples=200, noise=0.2, random_state=42)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

# Standardize the data
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train SVM with linear kernel
svm_linear = SVC(kernel='linear', probability=True, random_state=42)
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)

# Train SVM with RBF kernel
svm_rbf = SVC(kernel='rbf', gamma='scale', C=1.0, probability=True,␣
 ↪random_state=42)
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)

# Classification metrics function
def classification_metrics(y_true, y_pred, model_name):
    # Accuracy
    accuracy = accuracy_score(y_true, y_pred)

    # Classification report (includes precision, recall, f1-score)
    class_report = classification_report(y_true, y_pred)

    # Confusion Matrix for Specificity calculation
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    specificity = tn / (tn + fp)

    print(f"--- {model_name} ---")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Specificity: {specificity:.2f}")
    print("Classification Report:\n", class_report)

    return accuracy

# Plot ROC-AUC curve
def plot_roc_auc(svm_model, X_test, y_test, model_name):
    y_prob = svm_model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC-AUC Curve')
    plt.legend()
```

```
# Evaluate linear SVM
accuracy_linear = classification_metrics(y_test, y_pred_linear, "SVM with␣
 ↪Linear Kernel")
plot_roc_auc(svm_linear, X_test, y_test, "SVM Linear Kernel")

# Evaluate RBF SVM
accuracy_rbf = classification_metrics(y_test, y_pred_rbf, "SVM with RBF Kernel")
plot_roc_auc(svm_rbf, X_test, y_test, "SVM RBF Kernel")

# Show the ROC-AUC curves
plt.title("ROC-AUC Curve Comparison")
plt.show()
```

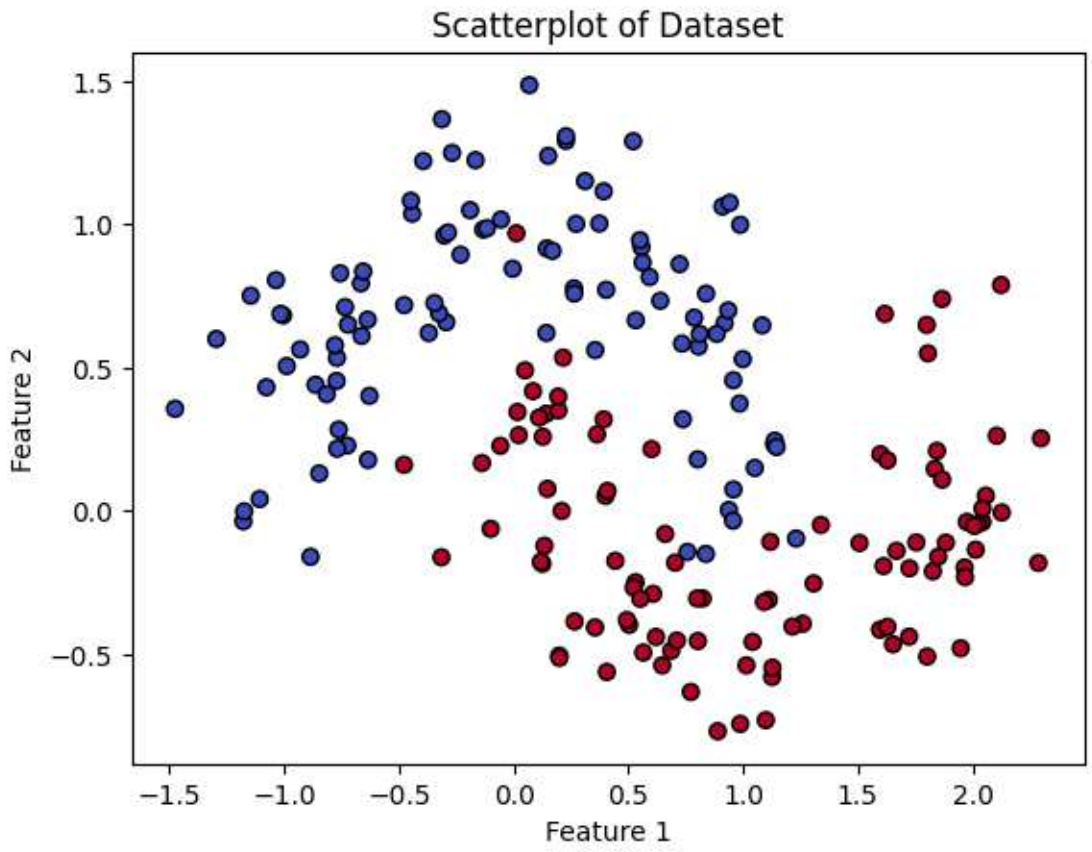
```
--- SVM with Linear Kernel ---
Accuracy: 83.33%
Specificity: 0.94
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.94      0.86        32
           1       0.91      0.71      0.80        28

    accuracy                           0.83        60
   macro avg       0.85      0.83      0.83        60
weighted avg       0.85      0.83      0.83        60

--- SVM with RBF Kernel ---
Accuracy: 98.33%
Specificity: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        32
           1       1.00      0.96      0.98        28

    accuracy                           0.98        60
   macro avg       0.98      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60
```



[ ]:
```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, roc_curve,␣
 ↪auc, confusion_matrix

# Load dataset (e.g., moon dataset)
X, y = datasets.make_moons(n_samples=200, noise=0.2, random_state=42)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

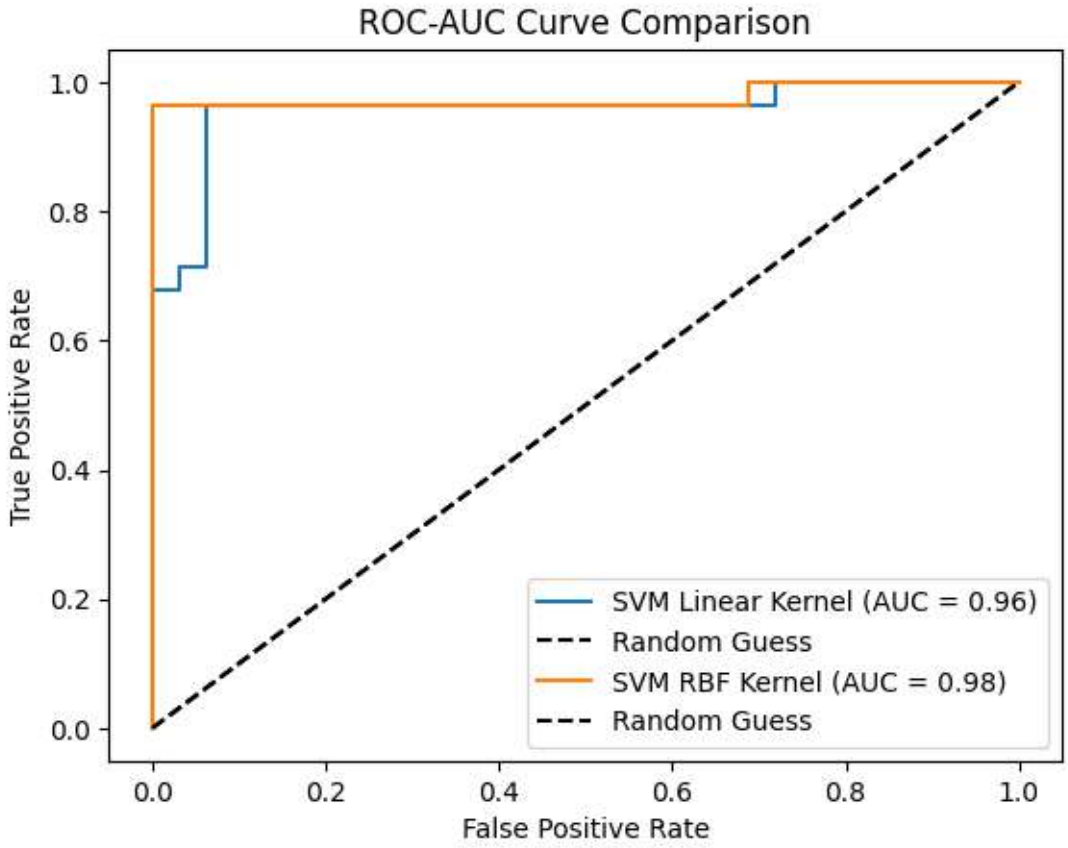
```
X_test = scaler.transform(X_test)

# Train SVM with linear kernel
svm_linear = SVC(kernel='linear', probability=True, random_state=42)
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)

# Train SVM with RBF kernel
svm_rbf = SVC(kernel='rbf', gamma='scale', C=1.0, probability=True,␣
 ↪random_state=42)
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)

# Classification metrics function
def classification_metrics(y_true, y_pred, model_name):
    # Accuracy
    accuracy = accuracy_score(y_true, y_pred)

    # Classification report (includes precision, recall, f1-score)
    class_report = classification_report(y_true, y_pred)


    # Confusion Matrix for Specificity calculation
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    specificity = tn / (tn + fp)

    print(f"--- {model_name} ---")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Specificity: {specificity:.2f}")
    print("Classification Report:\n", class_report)

    return accuracy

# Plot ROC-AUC curve
def plot_roc_auc(svm_model, X_test, y_test, model_name):
    y_prob = svm_model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC-AUC Curve')
    plt.legend()

# Scatterplot of the dataset
```

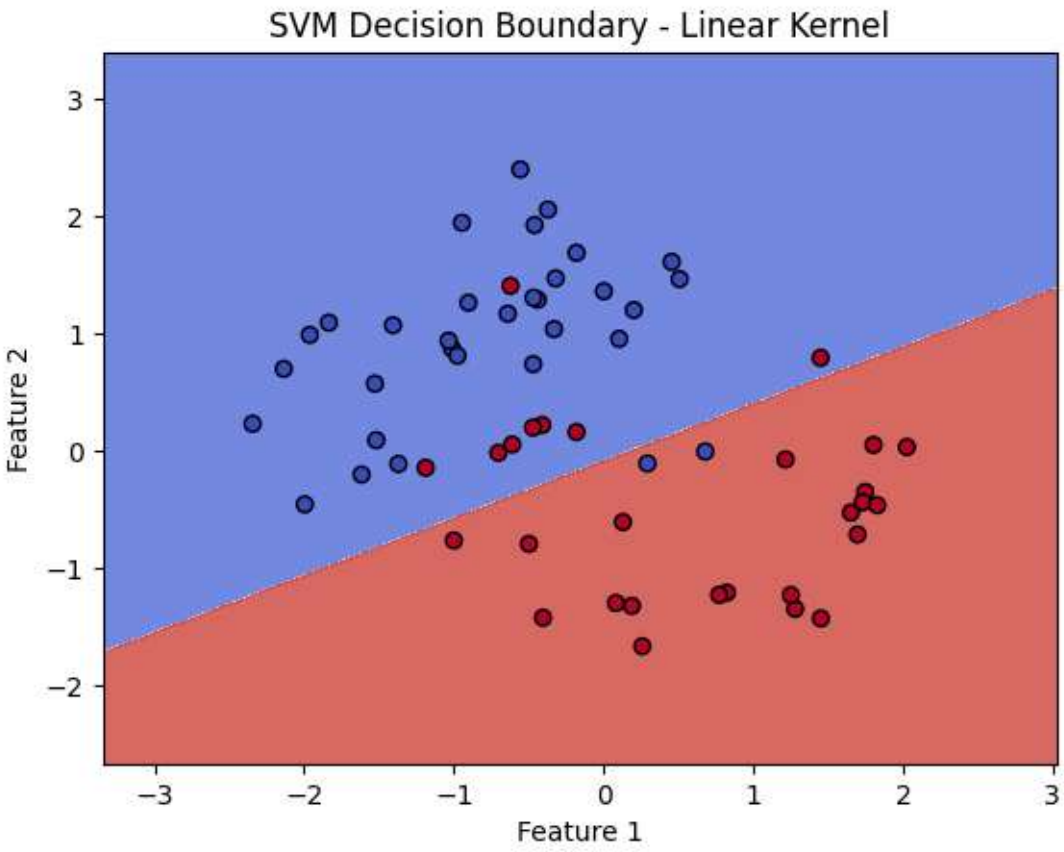
```
def plot_data(X, y):
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap='coolwarm', edgecolor='k')
    plt.title('Scatterplot of Dataset')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()

# Visualize the scatterplot
plot_data(X, y)

# Evaluate linear SVM
accuracy_linear = classification_metrics(y_test, y_pred_linear, "SVM with␣
 ↪Linear Kernel")
plot_roc_auc(svm_linear, X_test, y_test, "SVM Linear Kernel")

# Evaluate RBF SVM
accuracy_rbf = classification_metrics(y_test, y_pred_rbf, "SVM with RBF Kernel")
plot_roc_auc(svm_rbf, X_test, y_test, "SVM RBF Kernel")

# Show the ROC-AUC curves
plt.title("ROC-AUC Curve Comparison")
plt.show()

# Function to visualize decision boundaries
def plot_decision_boundary(clf, X, y, model_name):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max,␣
 ↪0.01))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', cmap=plt.cm.coolwarm)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title(f'SVM Decision Boundary - {model_name}')
    plt.show()

# Visualize decision boundaries
plot_decision_boundary(svm_linear, X_test, y_test, "Linear Kernel")
plot_decision_boundary(svm_rbf, X_test, y_test, "RBF Kernel")
```

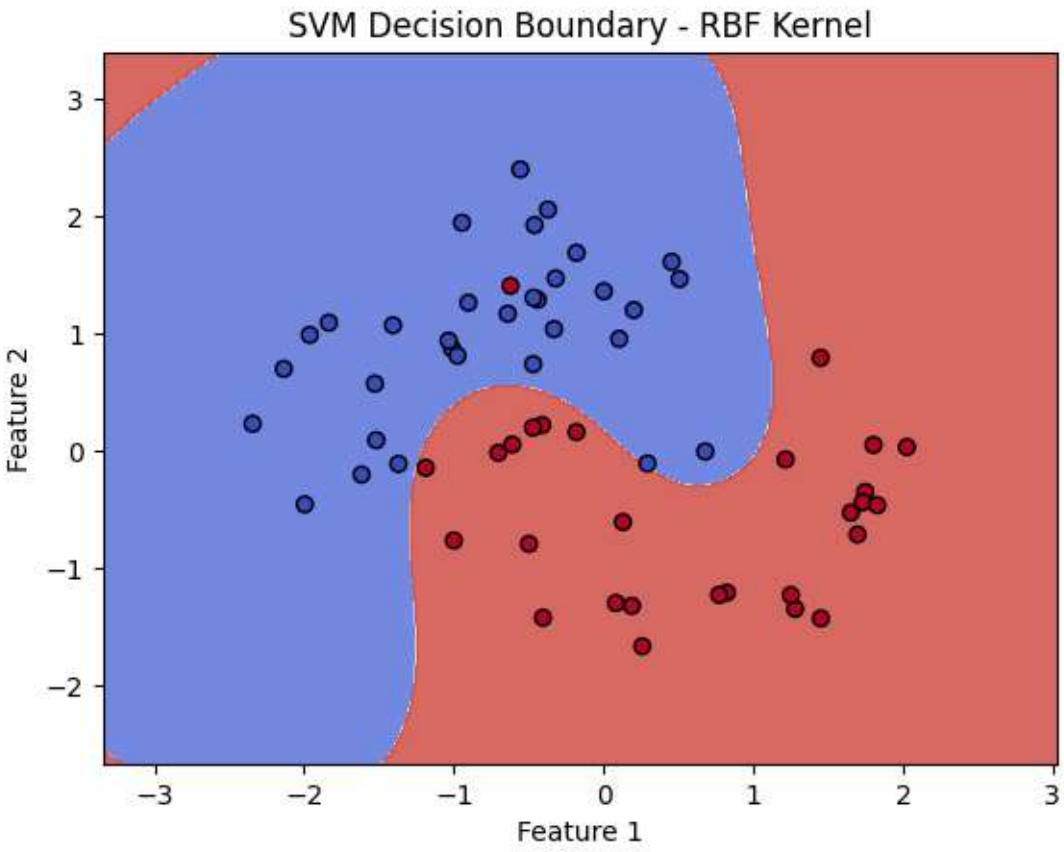



```
--- SVM with Linear Kernel ---
Accuracy: 83.33%
Specificity: 0.94
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.94      0.86        32
           1       0.91      0.71      0.80        28

    accuracy                           0.83        60
   macro avg       0.85      0.83      0.83        60
weighted avg       0.85      0.83      0.83        60

--- SVM with RBF Kernel ---
Accuracy: 98.33%
Specificity: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        32
```

1	1.00	0.96	0.98	28
accuracy			0.98	60
macro avg	0.98	0.98	0.98	60
weighted avg	0.98	0.98	0.98	60




SVM Decision Boundary - Linear Kernel
Feature 1
Feature 2
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


SVM Decision Boundary - RBF Kernel
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

ml-exp-7

October 18, 2024

[ ]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

[ ]:
```
# Import necessary libraries
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define base models
```

```
base_models = [
    ('decision_tree', DecisionTreeClassifier(random_state=42)),
    ('svm', SVC(probability=True, random_state=42))
]

# Define the stacking ensemble model
stacking_clf = StackingClassifier(
    estimators=base_models,
    final_estimator=LogisticRegression(),
    cv=5
)

# Train the stacking classifier
stacking_clf.fit(X_train, y_train)

# Make predictions and evaluate the accuracy
y_pred = stacking_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Ensemble Model Accuracy: {accuracy * 100:.2f}%")
```

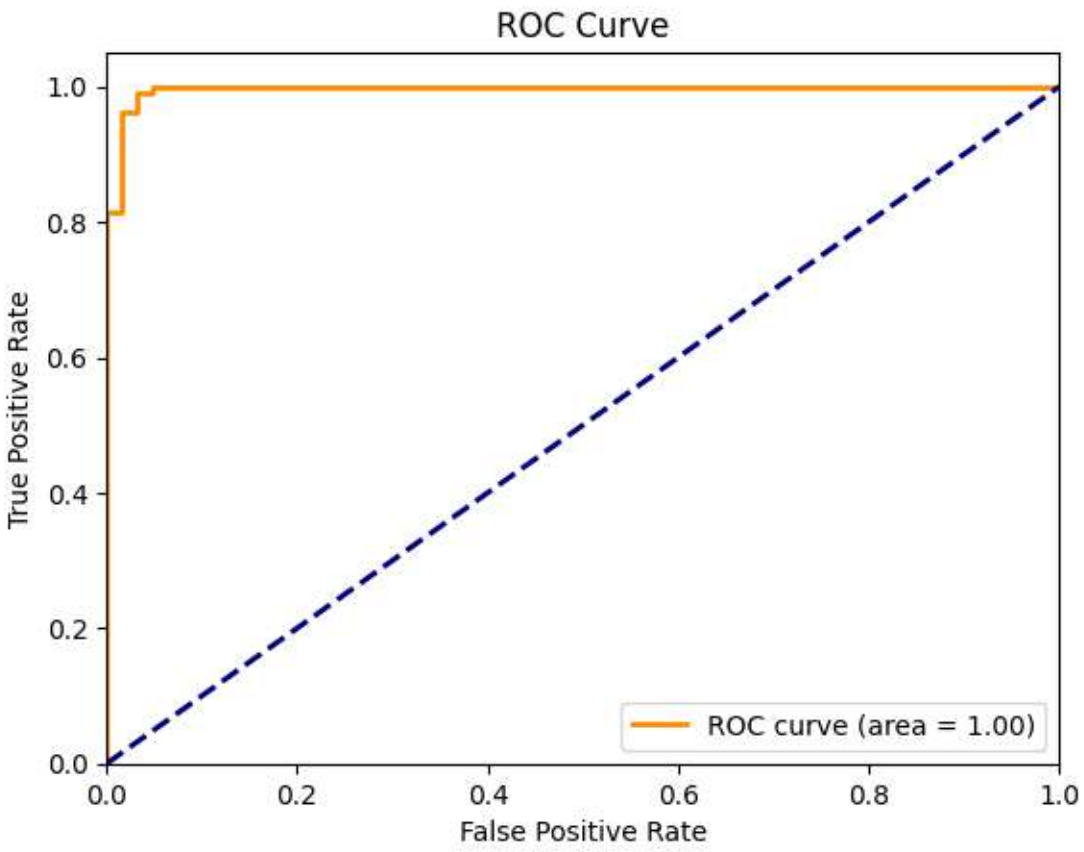
```
Ensemble Model Accuracy: 97.66%
```

[ ]:
```
# Import necessary libraries
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score,␣
 ↪roc_auc_score, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_predict

# Load the dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)

# Define base models for stacking
base_models = [
    ('decision_tree', DecisionTreeClassifier(random_state=42)),
    ('svm', SVC(probability=True, random_state=42))
]

# Define the stacking ensemble model
stacking_clf = StackingClassifier(
    estimators=base_models,
    final_estimator=LogisticRegression(),
    cv=5
)

# Train the stacking classifier
stacking_clf.fit(X_train, y_train)

# Make predictions and evaluate the accuracy
y_pred = stacking_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Ensemble Model Accuracy: {accuracy * 100:.2f}%")

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion matrix for specificity calculation
conf_matrix = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf_matrix.ravel()

# Specificity calculation
specificity = tn / (tn + fp)
print(f"Specificity: {specificity:.2f}")

# ROC-AUC score
y_pred_prob = stacking_clf.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_prob)
print(f"ROC-AUC Score: {roc_auc:.2f}")

# Plot ROC curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:
  ↪.2f})')
```

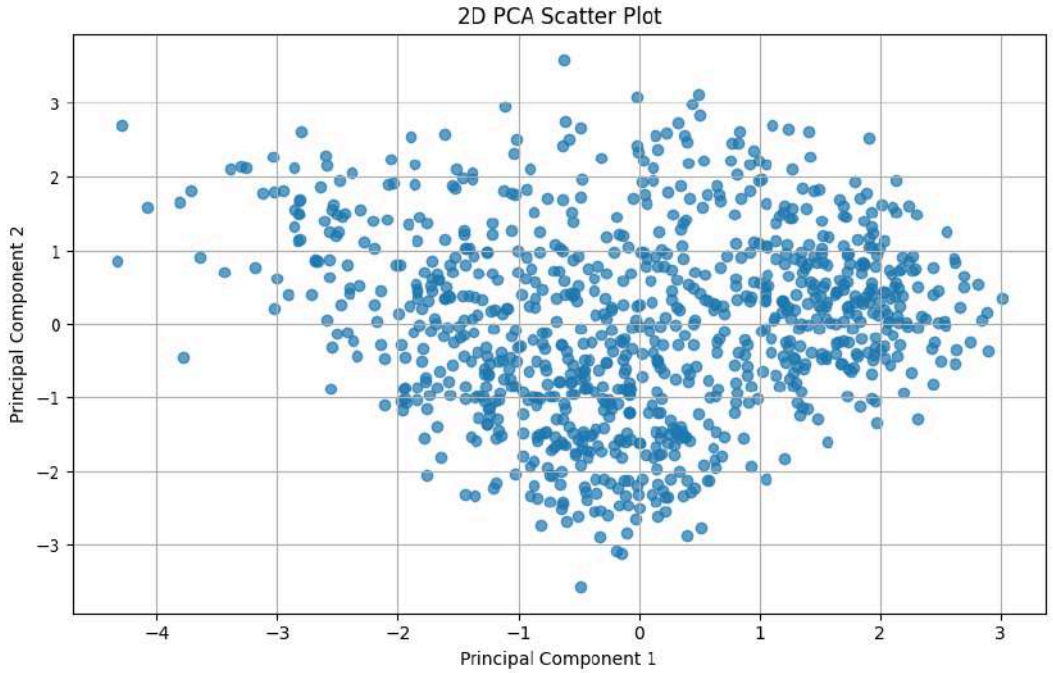
```
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()

# Final voting method: Get cross-validated predictions to demonstrate model␣
 ↪performance
print("\nFinal Voting Method (Cross-Validated Predictions):")
y_pred_cv = cross_val_predict(stacking_clf, X_train, y_train, cv=5)
print(f"Cross-Validated Accuracy: {accuracy_score(y_train, y_pred_cv) * 100:.
 ↪2f}%")
```

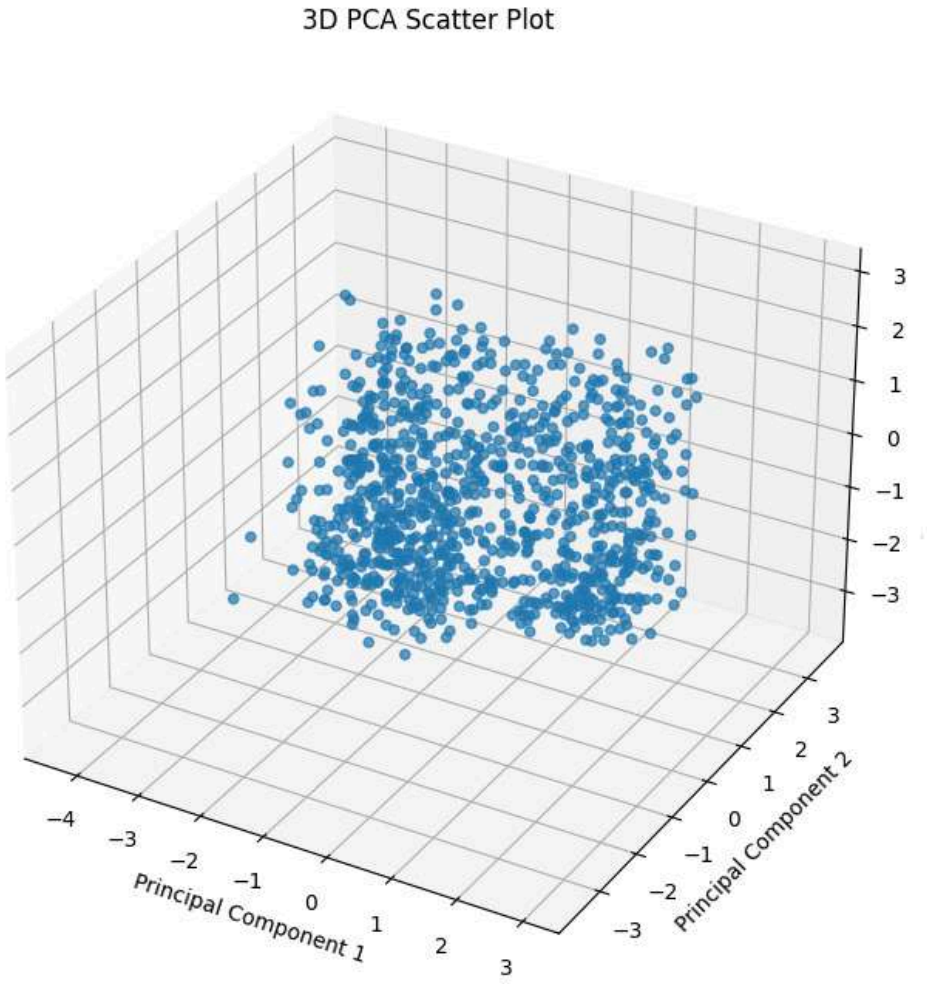
```
Ensemble Model Accuracy: 97.66%

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        63
           1       0.98      0.98      0.98       108

    accuracy                           0.98       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171

Specificity: 0.97
ROC-AUC Score: 1.00
```




```
Final Voting Method (Cross-Validated Predictions):
Cross-Validated Accuracy: 96.98%
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D

file_path = '/content/drive/MyDrive/kaggle/input/machine-failure.csv'
data = pd.read_csv(file_path)

features = ['footfall', 'tempMode', 'AQ', 'USS', 'CS', 'VOC', 'RP', 'IP', 'Temperature']
X = data[features]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(10, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.7)
plt.title('2D PCA Scatter Plot')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid()
plt.show()

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], alpha=0.7)
ax.set_title('3D PCA Scatter Plot')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
plt.show()
```


2D PCA Scatter Plot
Principal Component 2
Principal Component 1
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3


3D PCA Scatter Plot
Principal Component 1
Principal Component 2
−4
−3
−2
−1
0
1
2
3

Title: "Machine Learning in Detecting Air Quality Index"

Summary:

This case study examines the application of machine learning (ML) techniques in detecting and predicting the Air Quality Index (AQI). The AQI is a critical measure of air quality, reflecting the level of pollutants in the air and its potential impact on human health and the environment. By leveraging various ML algorithms, researchers and practitioners can enhance the accuracy of AQI predictions, facilitate timely alerts, and support informed decision-making for public health and environmental policies.

Introduction and Background:

The Air Quality Index (AQI) is a standardized indicator used globally to communicate air quality status to the public. It quantifies levels of common air pollutants such as particulate matter (PM), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), and ozone (O3). Machine learning, a subset of artificial intelligence, offers powerful tools for analyzing large datasets and making predictions. This section introduces fundamental concepts in ML, including supervised and unsupervised learning, regression models, and classification techniques, setting the foundation for their application in air quality monitoring.

Challenges:

Several challenges hinder the effective implementation of machine learning techniques in AQI detection:

- **Data Quality and Availability**: Incomplete, inconsistent, or low-quality data can significantly affect model performance. Limited access to high-resolution air quality data can also hinder accurate predictions.
- **Feature Selection**: Identifying relevant features that influence air quality can be complex. Factors such as weather conditions, geographical location, and industrial activities must be considered.
- **Model Interpretability**: Many advanced ML models, particularly deep learning algorithms, operate as "black boxes," making it difficult to interpret the results and understand the underlying mechanisms driving air quality changes.
- **Dynamic Nature of Air Quality**: Air quality can vary significantly over short periods and geographical areas, requiring models that can adapt to changing conditions.

Solutions and Techniques:

Various machine learning techniques and models are employed to address these challenges and improve AQI detection:

Traditional Time-Series Models:

- ARIMA (AutoRegressive Integrated Moving Average): Used for univariate time series forecasting, ARIMA models the relationship between current and past values, making it useful for predicting future AQI levels based on historical data.
- SARIMA (Seasonal ARIMA): An extension of ARIMA that accounts for seasonality in data, SARIMA is particularly effective in environments with seasonal patterns affecting air quality.

Machine Learning Models:

- Random Forest: This ensemble learning method combines multiple decision trees to improve prediction accuracy and robustness against overfitting. It can handle non-linear relationships and interactions between variables effectively.
- Support Vector Machines (SVM): SVMs can be used for classification tasks, categorizing air quality into different levels (e.g., good, moderate, unhealthy) based on input features.
- Neural Networks: Deep learning models can analyze high-dimensional data and identify complex patterns in air quality datasets, making them suitable for prediction tasks.

Data Preprocessing Techniques: Data cleaning, normalization, and transformation techniques are essential to improve data quality and model performance. Feature engineering can help identify and create relevant predictors.

Real-Time Monitoring and Prediction: Implementing real-time data collection through IoT devices and sensors enhances the ability to monitor air quality continuously and update predictions dynamically.

Trends:

Current trends in machine learning for AQI detection include:

- Integration of IoT and Big Data: The rise of IoT devices allows for the collection of vast amounts of real-time air quality data, providing a rich source for ML models.
- Ensemble Learning: Combining multiple models to improve accuracy and reliability in predictions is becoming increasingly popular, as it leverages the strengths of different algorithms.
- Explainable AI (XAI): As the need for transparency in AI applications grows, researchers are focusing on developing interpretable models that can provide insights into decision-making processes.
- Geospatial Analysis: Incorporating geospatial data and remote sensing technology enhances the ability to analyze air quality across different regions and contexts.

Applications in Real-World Scenarios:

Machine learning applications in AQI detection include:

- Predictive Modeling: Forecasting future AQI levels based on historical data and current environmental conditions, enabling proactive measures to mitigate air pollution.
- Health Impact Assessments: Analyzing the correlation between AQI levels and health outcomes to inform public health strategies and policies.
- Smart City Initiatives: Integrating air quality monitoring with urban planning and traffic management to create healthier urban environments.
- Public Awareness Tools: Developing applications that provide real-time AQI data and health advisories to the public, promoting awareness and encouraging protective behaviors..

Conclusion:

This case study highlights the transformative potential of machine learning in detecting and predicting the Air Quality Index. By addressing challenges, leveraging innovative solutions, and embracing current trends, ML techniques can significantly enhance the accuracy and effectiveness of air quality monitoring. As air quality remains a pressing global concern, continued research and development in this area will play a crucial role in safeguarding public health and the environment.

Group 38
Parth Dabholkar
C31
2103032

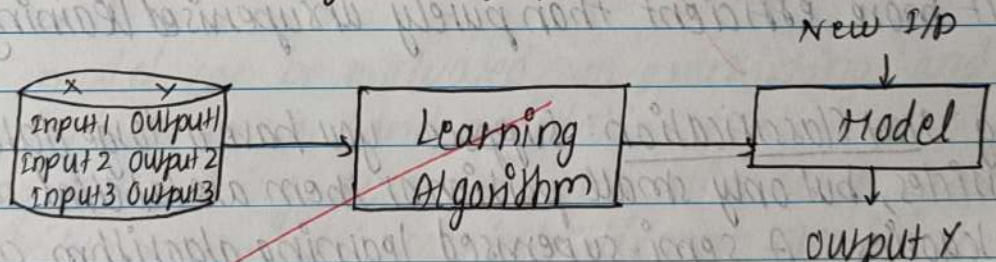
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO: 1

Q1] What is learning? Explain types of learning with example.

⇒ Machine Learning is a subset of Artificial Intelligence that enables algorithms to uncover hidden patterns within datasets, allowing them to make predictions on new similar data without explicit programming for each task. Machine Learning combines data with statistical tools to predict outputs, yielding actionable insights. This technology finds applications in diverse fields such as image and speech recognition, natural language processing, recommendation systems, fraud detection, portfolio optimization and automating tasks.

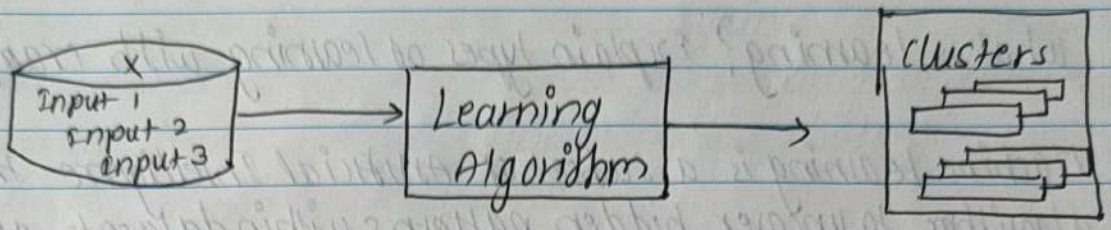
1) Supervised Learning: Supervised learning involves training a model on a labelled dataset, where the input data is paired with the correct output. The model learns to map inputs to the correct outputs by finding pattern in data.



Eg: Image classification: Given a dataset of labelled data images of cats and dogs, the model learns to classify new images either "cat" or "dog". Here the images are the input data, and the labels ("cat" or "dog") are the outputs the model is trained to predict.

2) Unsupervised Learning: Unsupervised learning involves training a model on data that does not have labelled outputs. The model finds

to find patterns or structures in the input data.



Unsupervised Learning

Eg: Customer Segmentation - A company may segment its customers into different groups based on purchasing behaviour. K-Means clustering can be used to cluster customers with similar purchasing history without any labels.

3) Semi-Supervised Learning: Semi-Supervised learning lies between supervised and unsupervised learning. It uses a small amount of labelled data and a large amount of unlabelled data. The labelled data helps in guiding the learning process, making it more efficient than purely unsupervised learning.

Eg: Text Classification: Suppose you have a large dataset of news articles, but only small portion of them are labelled with respective categories. A semi-supervised learning algorithm can use labelled articles to improve classification of unlabelled data.

4) Reinforcement Learning: Reinforcement Learning is a type of learning where an agent interacts with an environment and learns to make decisions by receiving feedback in the form of rewards and goals or penalties. The goal is to maximize the cumulative reward over time.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Eg: Game Playing AI: In training an AI to play chess, the algorithm learns by playing games against itself or other players. It recieves positive rewards for winning and negative rewards for losing, gradually improving its strategies to maximizes its chances of winning.

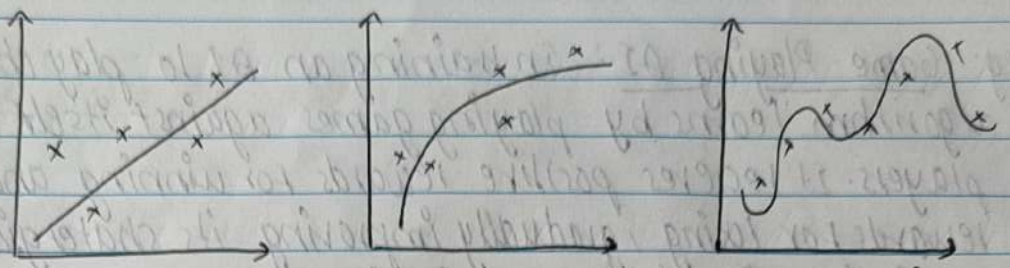
02] Define overfitting and underfitting. How to evaluate a ML model for overfitting or underfitting? Explain using diagrams. What measures are needed to be taken in-case of overfitting & underfitting?

⇒ Overfitting: It occurs when a training model learns the training data too well, capturing noise and details that do not generalize to new data. This leads to excellent performance on an unseen data.

Underfitting: It happens when a machine learning model is too simple to capture underlying patterns in the data. This results in poor performance on both training and unseen data.

A ML model can be evaluated for overfitting and underfitting using training error and validation error.
Plotting the training error and validation error against the complexity of the model can help diagnose overfitting and underfitting.
In understanding ~~both~~ ~~training~~, underfitting → training and validation errors are high. In overfitting, training error is low but validation error is high.
Good Fitting: Both training and validation errors are low and close to each other.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE



High Bias (Underfitting) | Low Bias, Low Error (Good Fitting) | High Variance (Overfitting)

Q3]- Illustrate the process of learning with the gradient descent for a linear regression using a bell shaped curve. Explain how a step size is modulated on every iteration.

⇒ Gradient descent is an iterative optimization algorithm that tries to find the optimum value (min/max) of an objective function. Gradient Descent is one of the most used optimization techniques in ML projects for updating the parameters of a model in order to minimize a cost function. In univariate linear regression, the model has a relationship between a single feature 'x' in a target variable of using Linear Regression equation:

$$y = \beta_0 + \beta_1 x$$

$\beta_1 \rightarrow$ Slope of the line

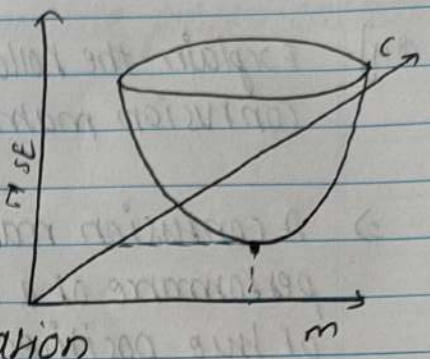
$\beta_0 \rightarrow$ Intercept

The cost/objective function is the error in our predicted value. We will use the Sum squared Error function to calculate the cost:

$$Q = \sum (y_i - (\beta_0 + \beta_1 x_i))^2$$

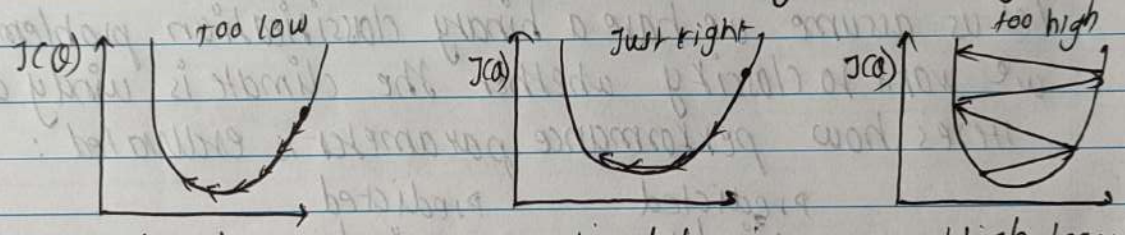
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

If we plot β_0 and β_1 against SSE = J(Q), it will aquire a bell shaped curve.



For some combination of β_0 and β_1, we will get least Error (MSE). That combination of β_0 and β_1 will give us our best fit line.

Step 1: Let $\beta_0 = 0$ and $\beta_1 = 0$. Let α be our learning rate. It could be a small value like 0.01 for good accuracy



small learning rate.

optimal learning rate

High learning rate.

Step 2: Calculate the partial derivative of cost function wrt β_0 and β_1.

$$\frac{\partial Q}{\partial \beta_0} = -2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) \qquad \frac{\partial Q}{\partial \beta_1} = -2\sum_{i=1}^{N} x_i(y_i - \beta_0 - \beta_1 x_i)$$

Step 3: Update the current value of β_0 and β_1 using following eqn:

$$\beta_0 = \beta_0 - \alpha\frac{\partial Q}{\partial \beta_0} \qquad \beta_1 = \beta_1 - \alpha\frac{\partial Q}{\partial \beta_1}$$

Step 4: Repeat until our cost function is very small (ideally 0).

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q4] Explain the following performance parameters using a confusion matrix.

⇒ A confusion matrix is a table that is used to evaluate the performance of a classification model. It shows the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) for the model. These values are the foundation for calculating several important performance metrics.

Let us assume we have a binary classification problem where we want to classify whether the climate is windy or not. Here's how performance parameter is evaluated:

	Predicted windy	Predicted Not windy
Actual windy	70	30
Actual ! windy	10	90

TP = 70 FN = 30
FP = 10 TN = 90

1. Accuracy: Measure the overall correctness of the model indicating how often the model's prediction are correct.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Eg: If model predicts 70 times that climate is windy and 90 as not windy, and incorrectly classifies 10 times as windy and 30 times not windy then:

$$\text{Accuracy} = \frac{70 + 90}{70 + 90 + 30 + 10} = \frac{160}{200}$$

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2. Precision: Measures how many of the positive predictions made by the model are actually correct:

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Precision} = \frac{70}{70+10} = \frac{70}{80}$$

3) Recall: Measures how many actual positive cases were correctly identified by the model. It focuses on model's ability to detect all positive instances.

Formula: $\text{Recall} = \frac{TP}{TP+FN}$

$$\text{Recall} = \frac{70}{70+30} = \frac{70}{100}$$

4) F1 - Score: It is the harmonic mean of precision and recall. It balances the two, especially useful when you need to account for both precision and recall.

Formula: $\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

$$\text{F1-Score} = \frac{2 \times 70/80 \times 70/100}{70/80 + 70/100} = 0.778$$

5) Specificity: Measures the proportion of actual negatives that were correctly identified. It tells us how well the model can identify negative instances.

Specificity: $\frac{TN}{TN+FP}$

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

$$\text{Specificity} = \frac{90}{90+10} = \frac{90}{100}$$

6) ROC-AUC curve: It is a graphical representation of the tradeoff between true positive rate (Recall) and false positive rate (FPR) across different threshold values. The AUC quantifies the overall ability of the model to discriminate between positive and negative values.

$$FPR = \frac{FP}{FP+N}$$

Interpreting AUC:

AUC = 1.0 Perfect classifier
AUC ≈ 0.5 No discrimination ability
AUC < 0.5 Worse than randomly guessing.

Eg: If the ROC-AUC curve for our wind detection model has an AUC of 0.84, it means the model has 84% chance of distinguishing between windy and not windy day correctly.

A+ 4/9/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO: 2

Q1] Discuss different ensemble learning techniques.

=> Ensemble learning is a machine learning technique that combines the predictions from multiple individual models to obtain a better predective performance than any single model. The basic idea behind ensemble learning is to leverage the wisdom of the crowd by aggregating the predictions of multiple models, each of which may have its own strengths and weaknesses. This can lead to improved performance and generalization.

Most commonly used ensembles includes techniques such as Bagging - used to generate Random Forest algorithms and Boosting - to generate algorithms such as Adaboost, Xgboost etc.

i) Bagging: Bagging involves training multiple models independently on different random subsets of the data (with replacement) and then aggregating their predictions, usually by averaging in regression tasks or voting in classification. Random Forest algorithm is a popular bagging method where many decision trees are trained on random subsets of data.
Advantages: Reduces variance and helps in preventing overfitting.
Disadvantages: Lead to large model sizes and requires more computational resources.

ii) Boosting: Boosting is a sequential ensemble techniques where models are trained one after another, each trying

to correct errors made by the previous models. The final prediction is a weighted sum of the predictions of all models. Adaboost, Gradient Boosting, XGBoost, Catboost etc. are sample key algorithms in Boosting.

Advantage: Can capture diverse patterns in data by combining different types of models.

Disadvantage: More complex to implement and tune, and requires careful validation to avoid overfitting.

iii) Stacking: Stacking involves training different models on the same dataset, then using a meta-model to combine their predictions. The base models' predictions are used as input features for the meta-model.

Advantage: Capture diverse patterns in the data by combining different types of models.

Disadvantage: More complex to implement and tune, and requires careful validation to avoid overfitting.

ii) Voting: Voting is a simple ensemble learning technique where multiple models are trained independently, and their predictions are combined by majority vote or by averaging.

Advantages: Easy to implement and often improve model performance.

Disadvantages: Less sophisticated and may not perform as well as boosting or stacking.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q2] - Explain the following terms:

a) Weak Learners: A weak learner is a model that performs slightly better than random guessing. It's typically a simple model with it's limited predictive power. often making predictions that are only marginally better than random chance.

Eg: In Decision tree algorithms, a shallow tree with a few nodes or a simple rule based classifier can be considered a weak learner.

Strong Learner: A strong learner is a model that performs significantly better than random guessing. It has high predictive accuracy and can capture complex patterns in the data.

Eg: Complex models like deep learning neural networks or ensemble methods like random forests are example of strong learnings.

b) Meta Learning: It refers to the process of learning how to learn. It involves developing models that can learn from various learning algorithms or datasets and adapt to new tasks more efficiently. The idea is to improve the learning process itself by using previous experience or data. Eg: Meta-Learning can involve learning the best hyper parameters for different algorithms selecting appropriate models for various tasks or even adapting to new types or data with minimal additional training.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2) Random Forest: It is an ensemble learning method specifically designed for classification and regression forests tasks. It builds a large no. of decisions tree and combines their predictions to make a final decision.

Working: Bagging - Random forests uses the bagging technique to train each decision tree on a different random subset of the training data sampled.

Feature-randomness: During the training of each decision tree, only a random subset of features is consistent for splitting at each node.

Aggregation: For classification tasks the final prediction is determined by majority voting among all the decision trees. For regression tasks, the final prediction is the average of all the predictions from all the trees.

3) Compare Bagging, Boosting and stacking.

⇒

Feature	Bagging	Boosting	stacking
i) Basic Concept	Train multiple models independently on random subsets of the data and aggregate their predictions	Train models sequentially, each correcting errors of the previous one.	Train multiple models and combine their predictions using a meta-model.
ii) Model Dependency	Independent	Sequential	Independent

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

iii) Goal	Reduce variance and prevent overfitting.	Reduce Bias and improve overall accuracy.	Capture diverse patterns by combining diff. models.
iv) Common Algorithms	Random Forest.	AdaBoost, Gradient Boosting, XGBoost.	Various models combined.
v) Complexity	Simple to implement and parallelizable.	More complex involves sequential and careful training.	Most complex involves training multiple models.
vi) Performance	Improves accuracy by reducing variance.	Improves accuracy by reducing both bias and variance.	Significantly improves performance.

Q4) Explain the AdaBoost Algorithm.

⇒ AdaBoost short for ~~Adaptive~~ Adaptive Boosting is an ensemble learning used in machine learning for classification and regression platforms. The main idea behind Adaboost is to iteratively train the weak classifier on the training dataset with each successive classifier giving more weightage to data-points that are mis-classified.
The final AdaBoost model is decided by combining all the weak classifier that has been used for training with the weightage given to the models according to the accuracies

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Algorithm:

Step 1: Initialize weights
- For datasets with N training data point instances, initialize 'N' wi weights for each data point $wi = 1/N$

Step 2: Train weak classifier
- Train weak classifier Mk where k is the current iteration.
- Accuracy should be greater than 0.5.

Step 3: Calculate the error rate and importance of each model α-k using formula $\alpha k = \frac{1}{2} \ln \frac{1 - \text{error}}{\text{error}}$

Step 4: Update data point weight for each data-point wi.
- Formula for updating will be:

$$wi = wi \exp(-\alpha k \, yi \, Mk(xi))$$

Step 5: Normalize the instance weight.
- They are summed up to 1 using formula:

$$wi = wi / Sum(w)$$

Step 6: Repeat steps 2-5 for weak iterations.
- We will train k classifiers and will calculate model importance and update the instance using the above formula.

A+ 5/4/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ML Assignment 3

Q1] Compare Decision tree classification with KNN classification.

Decision Tree classification	KNN Classification
Tree like model with nodes representing features and branches representing decisions.	Supervised learning Instance based learning method.
Faster to train, builds the tree by splitting the dataset recursively	Slower to train, involves storing entire set.
Faster prediction as it involves traversing the tree.	Slower prediction as it requires computing distance k.
Highly interpretable, the decision making process is easily visualized.	Less interpretable decision based on neighbours can be difficult to explain.
Sensitive to noisy data, overfitting is a common issue.	Sensitive to noisy data, especially if k is small.
Low memory usage once the tree is built.	High memory usage, store the entire training dataset.

Q2] Compare decision tree classification with logistic regression classification.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Decision tree classification	Logistic Regression classification
Tree like model with nodes representing features and branches representing decision.	Linear model that predicts probabilities using a logistic function.
Faster to train involves recursive data splitting.	Typically faster especially for small to medium datasets.
Faster prediction as it involves ~~train~~ traversing the tree.	Very fast prediction just a dot product of weights and features.
Can handle both categorized and numerical data.	Works best with numerical data, categorical needs to be scaled for optimized performance.
Not required, works with raw data.	Required. Feature needs to be scaled for optimized performance.
Low memory usage once the tree is built.	Very low memory usage, just stores coefficient.

03]- List down the attribute selection measures used by ID3 algorithm to construct a decision tree.

⇒ The ID3 algorithm constructs a decision tree by selecting attributes based on specific measures that help determine

best attribute for splitting the data at each node.

1) Entropy: Entropy is a measure of the uncertainty or impurity in a dataset. It quantifies the amount of disorder or randomness in the information being processed.

For a binary classification, entropy is calculated as:

$$\text{Entropy}(E) = -p_1 \log_2(P_1) - p_2 \log_2(P_2)$$

where P_1, P_2 are the properties of two classes in the dataset and S

2) Information Gain: Information Gain measures the reduction in entropy after the dataset is split on an attribute. It tells us how much information a particular attribute gives us about the class label.

Information gain is calculated as:

$$\text{Info gain}(S_A) = \text{Entropy}(S) - \sum_{i=\text{values}} \frac{|S_v|}{(S)} \cdot x \ \text{entropy}(S_v)$$

where S is the original dataset, S_v is the subset.

3) Gain Ratio. Gain ratio is a modification of information gain that reduces its bias attributes with many values. It adjusts information gain by considering the intrinsic information of a split

$$\text{Gain ration}(A) = \frac{\text{Information Gain}(S, A)}{\text{Intrinsic Information}(A)}$$

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

4) Gini Index: The Gini Index measures the impurity of a dataset and is used to determine the best attribute for splitting. It focusses on how often a randomly chosen element from the dataset would be incorrectly labelled if it were randomly labelled according to the distribution of labels in the dataset.

Gini Index is calculated as: Gini Index

$$S = 1 - \sum_{i=1}^{C} P_i^2$$

Q4) - Explain Properties of Gini Index.

⇒ The Gini Index is a measure of inequality or impurity of a distribution, commonly used in decision trees and other machine learning algorithm. It ranges from 0 to 0.5 indicates a maximally impure set (instances are evenly distributed across classes).

1). It is calculated by summing the squared probabilities of each outcome in a distribution and subtracting the result from 1.

2) A lower Gini Index indicates homogenous or pure distribution, while a higher Gini Index indicates a more heterogenous or impure distribution.

3) In decision tree, the Gini Index is used to evaluate the quality of a split by measuring the difference b/w impurity.

of parent node and weighted impurity of child node.

5) One disadvantage of Gini Index is that it tends to favor splits that create equally sized child nodes even when they are not optimal.

Gini Index method can be modified for categorized attributes. Gini is used in classification and regression tree (CART).

$$Gini(t) = 1 - \sum_{j=1}^{n} (P_j)^2$$

In above eqn P_j represents the relative frequency of class j in t.

$$Gini(split) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

Q5) Discuss in brief pruning in decision trees.

⇒ Decision tree pruning is a technique used to prevent decision trees from overfitting the training data. Pruning aims to simplify the decision tree by removing parts of it that do not provide significant predictive power, thus improving its ability to generalize new data.

Decision tree pruning removes unwanted nodes from the overfitted decision tree to make it smaller in size which results in more fast, more accurate and more effective predictions.

Types of Decision Pruning :

Pre-Pruning : Sometimes the growth of decision trees can be stopped before it gets too complex. This is called pre-pruning. It is important to prevent the overfitting of the training data, which results in a poor performance when exposed to new data.

Eg: 1) Maximum Depth: It limits the maximum level of depth in a decision tree.
2) Maximum samples per leaf.
3) Minimum samples per split.
4) Maximum features.

Post-Pruning : After the tree is fully grown, post pruning involves removing branches or nodes to improve the model's ability to generalize. Some common post pruning techniques are:

1) Cost complexity pruning: Assigns price to each subtree primarily based on its accuracy and complexity.

2) Reduced error pruning : Removes branches that do not significantly affect the overall accuracy.

3) Minimum Leaf size : Prunes nodes if the decrease in impurity is beneath a certain threshold.

4) Minimum Leaf size : Removes leaf nodes with fewer samples.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ML Assignment no: 4

Q1] Explain the intuition behind logistic regression in detail. Is decision boundary linear or non-linear in case of logistic regression model. Explain the impact of outliers on logistic Regression.

⇒ Logistic Regression is a classification algorithm used to predict the probability of binary outcome (0 or 1). It first calculates a linear combination of i/p features, then applies the sigmoid function to map this result to a probability between 0 and 1. Based on this probability, the model classifies the input into one of two categories, typically using a threshold of 0.5.

Linear combination of features: Logistic regression starts similarly to linear regression, where we compute weighted sum of i/p features plus a bias term (intercept).

$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n + b.$$

x_1, x_2, x_n are i/p features, $w_1, w_2 \dots w_n$ are weights and b is the bias term.

Sigmoid Function: Instead of predicting the output directly like linear regression, logistic regression passes the linear combination z through the sigmoid function to map it into the range [0, 1], representing the probability of +ve cases.

$$\hat{y} = 1 / 1 + e^{-z}$$

Decision Boundary: Linear or Non Linear: In its basic form logistic regression has a linear decision boundary. It seperates classes using a straight line. However if you include non-linear features, logistic regression can create a non-linear boundary.

Impact of outliers: Outliers can negatively impact logistic regression by skewing the model coefficients, leading to poor classification and distorted probabilities. Techniques like regularization or outlier removal can help mitigate this issue.

Q2] What assumptions are made in Logistic Regression? Can we solve multiclass classification problems using Logistic Regression? If yes then how?

⇒ 1) Linear Relationship: Logistic regression assumes a linear relationship between the independent variable and the log odds of the dependent variables. However this is a linearity in log odds, not in actual dependent variable.

2) Independent Observation: The observations should be independent of each other, meaning there should be no auto-correlation and relationship b/w individual data-points.

3) No multi-collinearity: Logistic regression assumes that there is no point in perfect multicollinearity among the independent variables. High collinearity between predictors can lead to unreliable estimate the model parameters.

4) Binary Outcome: The dependent variable is binary (0 or 1) This assumption holds for multiclass classification, we use extensions of logistic regression.

5) Large sample size: Logistic Regression works best with a larger sample size, especially when there are many predictors. This helps ensure stability of the coefficient estimates

Yes, logistic regression can handle multiclass classification problems using the following techniques:

1) One-Vs-Rest (OVR) or One-Vs-All (OvA) Approach:-
→ In this method, a seperate binary classifier is trained for each class.
→ For each classifier, one class is considered the "positive" class and all other classes are grouped into "negative" class.
→ During Prediction, each classifier outputs a probability score, and the class with highest probability is chosen.
→ This approach is simple and efficient for most cases but many struggle if classes are not well-seperated.

2) Multinomial Logistic Regression:
→ Instead of building multiple binary classifiers, multinomial logistic regression is directly models the probability of each class, using the softmax function.
→ Softmax function is an extension of the signed function used to compute probabilities for more than two classes. The formula is : $P(y=c_k|x) = e^{z_k} / \sum_{j=1}^{C} e^{z_j}$
where z_k is the linear combination of $J=1$ features for class K and C is the total number of classes.
→ The model predicts the class with the highest probability.
→ This approach is suited for handling inherently multiclass problems compared to OVR.

03] Why is Logistic Regression termed as Regression and not as classification?

→ Logistic Regression is termed as 'regression' instead of "classification" because of its underlying mathematical approach even though it is used for classification tasks.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Reasons: 1) Linear Regression Foundation:
Logistic Regression is based on the principle of linear regression. It starts by calculating a linear combination of input features.

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

The difference happens in further steps.

2) Regression of log-odds: Logistic Regression models the log-odds as a linear combination of the features. It is still performing regression but it's predicting the log-odds of binary outcome not the actual class label.

3) Continuos Probability Count: Instead of directly outputting a class (0 or 1), logistic regression computes a continuous probability (between 0 and 1) using the sigmoid function. The classification is a secondary step, where the probability is threshold (e.g ≥ 0.5 is classified as 1).

4) Naming Tradition: Historically, the term "logistic regression" stuck because it's fundamentally a regression technique to predict a continuous outcome. The classification part is derived from the probability, which is why it's not called classification.

4) Can we use Mean Square Error as a cost function for logistic regression? Justify your answer.

⇒ No, MSE is not typically used as a cost function for logistic regression.

1) Logistic Regression Output Probabilities: Logistic regression is used for classification tasks and predicts probabilities between 0 and 1. These probabilities are generated using the sigmoid function. MSE is designed for regression problems in where the output is continuous. Using MSE would result in inefficient optimization because it doesn't align well with the sigmoid output.

2) Non convexity of the cost function: When MSE is used with logistic regression, the resulting cost function becomes non-convex. This makes optimization difficult, as gradient based methods may get stuck in local minima and not coverage to the global minimum. In contrast, the log loss or binary cross-entropy function, which is typically used in logistic regression is convex and ensures a better and more reliable convergence.

3) Slow Convergence: Even if MSE were to be used, it would lead to slower convergence because the gradient updates would not be as effective in minimizing the error. The MSE does not penalize misclassifications in the same way as the log loss function, which is specifically designed for handling probability-based classification tasks.

Thus, the correct cost function for logistic regression is the log loss, as it better captures the essence of classification by measuring the distance between the predicted probabilities and actual binary outcomes.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q5] Compare naive Baiyes with logistic regression classifier in tabular form.

Naive Baiyes	Logistic Regression
i) Probabilistic classifier based on Baiye's Theorem.	i) Discriminative classifier based on linear models.
ii) Assume features are conditionally independent.	ii) No strong assumptions about feature independence.
iii) Non-linear in general, depends on the likelihood.	iii) Linear decision boundary.
iv) Predicts probabilities using Baiye's theorem.	iv) Predicts probabilities using the sigmoid function.
v) Works well with small datasets.	v) Requires more data for better performance.
vi) Suitable for text classification (Spam filtering, sentiment analysis).	vi) Suitable for binary classification problems (fraud detection, medical diagnosis.

A SB 7/10/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO: 5

Q1] Discuss the need of SVM. Explain why they are called as optimal binary classifiers.

⇒ Support Vector Machines are supervised ML algorithm used primarily for classification tasks, but also for regression. They are particularly effective due to:-

(i) Handles high-dimension data:-
⇒ works well with large feature spaces.

(ii) Maximum Margin:
⇒ Ensures decision boundary is far from data points, improving generalization.

(iii) Handles non-linear data:
⇒ Uses kernel functions to transform non-seperable data into linearly seperable space.

(iv) Resistant to overfitting:
⇒ Especially in high dimmensional spaces, due to margin maximization.

(v) Versatile:
SVMs can be trained with different functions and can use custom kernel functions.

SVMs also have a feature that can ignore outliers.

They are also called optimal binary classifiers because:-

(i) Maximizes Margin: Finds the hyperplane that maximizes the distance between classes.

(ii) Binary classifiers : seperates data into two different distinct classes.

(iii) optimal decision boundary : Ensures better performance by reducing risk of misclassification on new data.

Q2]. Explain the following with appropate illustration.

⇒ (1) optimal decision boundary :-

The optimal decision boundary in SVM is the one that maximizes the distance from nearest data points of each class. This is done to improve generalization.

A line that seperates two classes of points (circles and squares) while maximizing the distance from them.

(2) Support vectors :-

These are the data points that lie closest to the decision boundary from both the classes. These points are critical as they define the margin and position of optimal decision boundary. The nearest point from each of class to the boundary that influence its positions.

(3) Margins

The margin is the distance beweeen the decision boundary and the nearest data points to each class. SVM aims to maximizes these boundaries margins. A larger margin leads to better generalization.

The pardlel dashed lines on either side of the boundary that influence its position.

Q3] What do you mean by Linear and non-Linear Classifiers? Can you use SVM as non-linear classifier? If yes, explain how?

⇒ Linear Classifier:
A linear classifier seperates data using a straight line. It assumes that the classes can be seperate linearly in feature space. Eg: Logistic Regression or SVM with a linear kernel.
Non-Linear Classifier:
A non-linear classifier seperates data that cannot divided by a straight line. It uses more complex decision making boundaries to classify data. Eg: Decision tree, Neural Network.

Yes, SVM can be used as Non-Linear classifier by employing the kernel trick.
⇒ Kernel trick: SVM uses Kernel trick function to map the original non-linearly seperable data into a higher dimmensional space, where the data becomes linearly seperable. The SVM that finds a linear dimmension boundary in the higher dimmensional space.
Popular kernels:
Polynomial kernel: Maps data to a higher polynomial dimmension.
Radial Basis Function: Creates decision boundaries in a complex, curved shaped by mapping data to infinite dimmension.

Q4] Express the SVM constraint optimization problem. Discuss how prediction can be done using SVM. Support your answer with appropriate equations.
→ For linearly seperable dataset,

$$wx + b = 0$$

w: weight vector, x: i/p feature vector, b: bias term.
Constraints: $y_i(w_i x_i + b) \geq 1$ (x: data point, y - label)
If $y_i = +1$ → point on or above hyper plane.
$y_i = -1$ → point by on or below hyperplane.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Formulation: The primal form of SVM optimization is

$$\min \frac{1}{2} ||w||^2$$

subject to,

$$y_i (w_i x_i + b) \geq 1 \quad \forall i$$

Decision Formula: For new i/p xc:-

$$F(x) = wx + b$$

if $F(x) > 0$ point classified as +1

if $F(x) < 0$ point classified as -1.

Prediction Rule: Rule for new i/p x is $\bar{y} = sign(wx+b)$ assign class +1 or -1.

Support Vector Rule: w is computed as: $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

α_i = lagrange multiplier.

Thus, decision function becomes,

$$F(x) = \sum_{i=1}^{n} \alpha_i y_i (x_i x) + b.$$

This eqn shows that only support vectors x_i contribute to decision formula vectors x_i ~~can~~ making SVM seperate model.

Q5] what are kernel functions? List some kernel functions. Use in SVM?

=> Kernel functions are mathematical functions used in SVM to enable learning of non-linear decision boundaries by mapping the i/p data into a higher dimensional space.

- Kernel Functions: i) Linear kernel : $k(x, x') = x^T x'$
  use for linearly seperable data.
- Polynomial kernel : $k(x, x') = (x^T x' + c)^d$
  Allows modelling of polynomial relationships, where c & d are hyperparameters.

- Radial Basis Function (RBF):

$$K(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right)$$

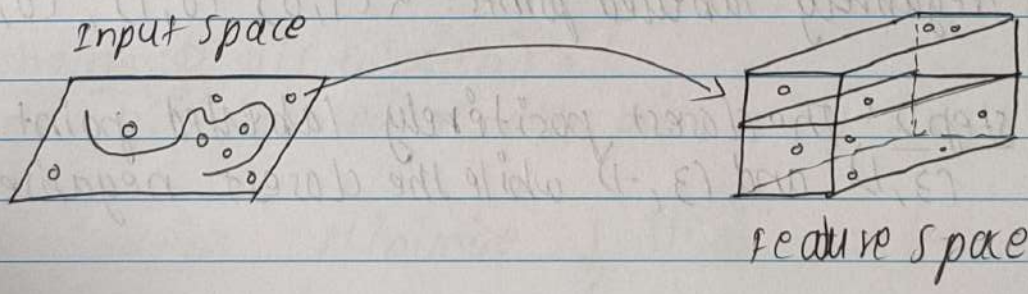
Effective for capturing non-linear patterns, where σ controls width or distribution.

- Sigmoid kernel: $K(x, x') = \tanh(\alpha x^T x' + c)$
  Similar to neural networks, where α and c are hyperparameters.

Use in SVM: i) Non-Linear Classification: Kernel transforms data into a higher-dimensional space where a linear hyperplane can be used to seperate classes.
ii) Computational Efficiency: Kernels allow the computation of inner products in the higher dimmensional space without explicitly mapping the data, using a technique called kernel trick.
iii) Flexibility: Different kernel functions offer the flexibility to chose the best one for a given dataset, making SVM adaptible to various types of data distribution.

Q6] Explain the concept of kernel trick. Discuss with example how kernel tricks can speed computation.

⇒ The kernel trick is a method used in SVM to handle non-linear data efficiently by computing inner products in a higher dimensional space without explicitly transforming the data.



Concept: (i) Implicit Mapping:
Kernels compute inner products in a higher-dimensional space directly.
(ii) Efficiency: Avoids the computational cost of explicit transformation.

Example: Polynomial kernel: $K(x, x') = (x^T x' + c)^d$
For $d = 2$, this computes:-
$K(x, x') = (x^T x' + c)^2$

Efficiency: This kernel function implicitly computes the higher dimensional feature needed to seperate non-linear data, speeding up computations by avoiding explicit feature transformations.

Q7] Given a '+'ly labelled data points as: {(3,1), (3,-1), (6,1), (6,-1)} and '-' labelled as: {(1,0), (0,1), (0,-1), (-1,0)}. Find parameters of decision boundary using SVM and classify (1,3).

Step 1: Visualizing the points.

Positively labelled points: {(3,1), (3,-1), (6,1), (6,-1)}

Negatively labelled points: {(1,0), (0,1), (0,-1), (-1,0)}

Step 2: The closest positively labelled point appears to (3,1) and (3,-1) while the closest negative is (1,0) (-1,0)

The decision boundary must lie somewhere between $x_i = 2$ between negative points and $x_i = 0$ and positive around $x_i = 3$.

The line seperating these points are having general form:

$$x_i = 2.$$

This is vertical line:

The eqn of this line is $x_i = 2$.

This means the weight factor $w = (1, 0)$ and bias $\boxed{b = -2}$

The decision boundary is $\boxed{x_i - 2 = 0}$

Step 3: To classify the point (1, 3) substituting its co-ordinates $(x_1, x_2) = (1, 3)$ into the equation $1 - 2 = -1$

Since the result is negative, the point (1,3) lies on negative side of boundary.

Classified as negative.

Obj- Obtain the optimal binary hyperplane for classifying the data points given below: {(1,1) (3,1) (1,4)} {(0,4) (3,3) (5,1)}

⇒ Positive (class (+1)) : {(1,1) (3,1) (1,4)}
Negative (class (−1)) : {(2,4) (3,3) (5,1)}

We want hyperplane defined by: $w^T x + b = 0$

→ Constraints: $y_i (w^T x_i + b) \geq 1$
where y_i is labelled of data point x_i.

→ Optimization: Minimize $\frac{1}{2} \|w\|^2$ subject to:-

$y_i(w^T x + b) \geq 1$ for $i = 1 \ldots 6$

we find that, • Weight factor: $w_i(1, -1)$

• Bias $b : -2$

Verifying the hyperplane:

For +ve points: $w^T x + b \geq 1$

For -ve points: $w^T x + b \leq -1$

→ Decision function is: $f(x) = w^T x + b$

$$f(x) = x_1 - x_2 - 2$$

→ The optimal hyperplane is Eqn: $x_1 - x_2 - 2 = 0$.

(A) 7/10/24