Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss Aarnav Malvia

of Computer Engg Department, Semester VII with

Roll No. 2103109 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 17/10/24

Principal

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	Select appropriate dataset and perform: i) Imputation ii) Anomaly detection iii) Standardisation iv) Normalisation v) Encoding		26/7/24	
2.	Implement gradient descent algorithm to minimize given objective function.		2/8/24	
3.	Implement Simple Linear Regression using analytical and machine learning method without using SKLearn.		9/8/24	
4.	Implement Logistic Regression using machine learning methods without using SKLearn.		16/8/24	
5.	Implement Decision tree algorithm.		23/8/24	
6.	Implement Support Vector Machines for non linear classification.		30/8/24	
7.	Implement ensemble methods to combine different models.		6/9/24	
8.	Implement principal component analysis technique.		13/9/24	
9.	Case Study		20/9/24	
10.	Assignment 1		12/7/24	
11	Assignment 2		19/7/24	
12	Assignment 3		27/9/24	
13	Assignment 4		4/10/24	
14	Assignment 5		11/10/24	

12/10/24

Experiment 1: Select appropriate dataset and perform following data preprocessing steps:

1. Imputation
2. Anomaly Detection
3. Standardization
4. Normalization
5. Encoding

```
import pandas as pd
from sklearn.impute import SimpleImputer
from scipy import stats
from sklearn.preprocessing import StandardScaler, MinMaxScaler

data = {
    'Student ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Exam 1 Score': [85, 76, 90, 65, 88, None, 78, 92, 85, 70],
    'Exam 2 Score': [92, 78, 88, 75, 91, 82, 76, 96, 89, 68],
    'Exam 3 Score': [88, None, 94, 80, 87, 79, 72, 98, 91, 75],
    'Final Grade': ['A', 'B', 'A', 'C', 'A', 'B', 'C', 'A', 'A', 'B']
}

df = pd.DataFrame(data)
print(df)
```

```
   Student ID  Exam 1 Score  Exam 2 Score  Exam 3 Score Final Grade
0           1          85.0            92          88.0           A
1           2          76.0            78           NaN           B
2           3          90.0            88          94.0           A
3           4          65.0            75          80.0           C
4           5          88.0            91          87.0           A
5           6           NaN            82          79.0           B
6           7          78.0            76          72.0           C
7           8          92.0            96          98.0           A
8           9          85.0            89          91.0           A
9          10          70.0            68          75.0           B
```

```
#imputation
imputer = SimpleImputer(strategy='mean')
df[['Exam 1 Score', 'Exam 2 Score', 'Exam 3 Score']] =
imputer.fit_transform(df[['Exam 1 Score', 'Exam 2 Score', 'Exam 3
Score']])

#anomaly detection
z_scores = stats.zscore(df[['Exam 1 Score', 'Exam 2 Score', 'Exam 3
Score']])
```

```
threshold = 3
outliers = (abs(z_scores) > threshold).any(axis=1)
df['Is Outlier'] = outliers

#standardization
scaler = StandardScaler()
df[['Exam 1 Score', 'Exam 2 Score', 'Exam 3 Score']] =
scaler.fit_transform(df[['Exam 1 Score', 'Exam 2 Score', 'Exam 3
Score']])

#normalization
min_max_scaler = MinMaxScaler()
df[['Exam 1 Score', 'Exam 2 Score', 'Exam 3 Score']] =
min_max_scaler.fit_transform(df[['Exam 1 Score', 'Exam 2 Score', 'Exam
3 Score']])

#output
print("Final Preprocessed Dataset:\n", df)
```

```
Final Preprocessed Dataset:
    Student ID  Exam 1 Score  Exam 2 Score  Exam 3 Score Final
Grade  \
0           1      0.740741      0.857143      0.615385           A
1           2      0.407407      0.357143      0.495726           B
2           3      0.925926      0.714286      0.846154           A
3           4      0.000000      0.250000      0.307692           C
4           5      0.851852      0.821429      0.576923           A
5           6      0.592593      0.500000      0.269231           B
6           7      0.481481      0.285714      0.000000           C
7           8      1.000000      1.000000      1.000000           A
8           9      0.740741      0.750000      0.730769           A
9          10      0.185185      0.000000      0.115385           B

   Is Outlier
0       False
1       False
2       False
3       False
4       False
5       False
6       False
7       False
8       False
9       False
```

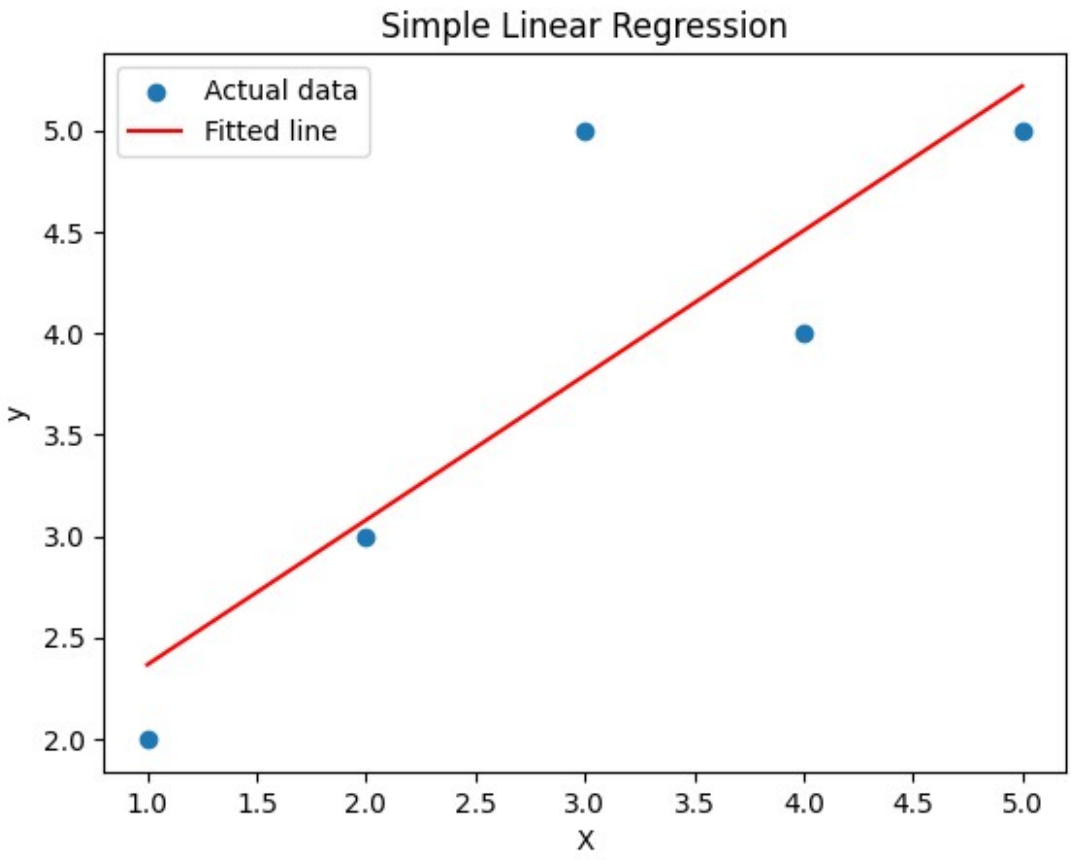
Experiment 2: Implement Gradient Descent Algorithm to minimize the given objective function

```
import numpy as np

#generate sample data
np.random.seed(42)
X = np.random.rand(100,1)
y = 3 * X + 2 + 0.1 * np.random.rand(100,1) #y=3x+2+noise

#init params
learning_rate = 0.01
epochs = 1000
weights = np.random.rand(2,1) #slope & intercept

print("Initial weights: ",weights)

#gradient descent
for epoch in range(epochs):
  X_b = np.c_[np.ones((len(X),1)),X] #add bias to X
  y_pred = X_b.dot(weights)
  error = y_pred - y
  gradients = 2 * X_b.T.dot(error) / len(X_b)
  weights -= learning_rate * gradients

  mse = np.mean(error ** 2)
  if epoch % 100 == 0:
    print(f"Epoch {epoch}, MSE: {mse}")

print("Final weights: ",weights)
```

```
Initial weights:  [[0.64203165]
 [0.08413996]]
Epoch 0, MSE: 8.465416355968795
Epoch 100, MSE: 0.2606143158871519
Epoch 200, MSE: 0.15588609482493906
Epoch 300, MSE: 0.11736675673739716
Epoch 400, MSE: 0.08861883059575491
Epoch 500, MSE: 0.06696513101893481
Epoch 600, MSE: 0.05065366014013287
Epoch 700, MSE: 0.03836641690206189
Epoch 800, MSE: 0.02911057796076158
Epoch 900, MSE: 0.02213826099330367
Final weights:  [[2.26626904]
 [2.57191926]]
```

Experiment 3: Implement Simple linear regression using analytical & ML method without using SKLearn package

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

class SimpleLinearRegression:
  def __init__ (self, learning_rate=0.01, num_iterations=1000):
    self.learning_rate = learning_rate
    self.num_iterations = num_iterations
    self.slope = None
    self.intercept = None

  def fit(self, X, y):
    num_samples, num_features = X.shape
    self.slope = np.zeros(num_features)
    self.intercept = 0
    for _ in range(self.num_iterations):
      y_pred = np.dot(X, self.slope) + self.intercept
      grad_slope = -(2 / num_samples) * np.dot(X.T, y - y_pred)
      grad_intercept = -(2 / num_samples) * np.sum(y - y_pred)
      self.slope -= self.learning_rate * grad_slope
      self.intercept -= self.learning_rate * grad_intercept

  def predict(self, X):
    return np.dot(X, self.slope) + self.intercept

#Sample Data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 3, 5, 4, 5])

model = SimpleLinearRegression(learning_rate=0.01,
num_iterations=1000)
model.fit(X, y)

y_pred = model.predict(X)

plt.scatter(X, y, label="Actual data")
plt.plot(X, y_pred, color='red', label="Fitted line")
plt.xlabel("X")
plt.ylabel("y")
plt.title("Simple Linear Regression")
plt.legend()
plt.show()
```




```
print("Slope:", model.slope[0])
print("Intercept:", model.intercept)
```

```
Slope: 0.7131035165747112
Intercept: 1.6526921474419984
```

Experiment 4: Implement Logistic regression using ML without using SKLearn

```
import numpy as np
import matplotlib.pyplot as plt
class LogisticRegression:
    def __init__ (self, learning_rate=0.01, num_iterations=1000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None

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

    def fit(self, X, y):
```

```
        num_samples, num_features = X.shape
        self.weights = np.zeros(num_features)
        self.bias = 0
        for _ in range(self.num_iterations):
            linear_model = np.dot(X, self.weights) + self.bias
            predictions = self.sigmoid(linear_model)
            dw = (1 / num_samples) * np.dot(X.T, (predictions - y))
            db = (1 / num_samples) * np.sum(predictions - y)
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db

    def predict(self, X):
        linear_model = np.dot(X, self.weights) + self.bias
        predictions = self.sigmoid(linear_model)
        predicted_labels = [1 if p >= 0.5 else 0 for p in predictions]
        return predicted_labels

# Sample data
X = np.array([[2.5, 1.5], [3.0, 1.0], [4.0, 3.0], [1.0, 4.0], [2.0,
2.0]])
y = np.array([1, 1, 0, 0, 1])

model = LogisticRegression(learning_rate=0.01, num_iterations=1000)
model.fit(X, y)

y_pred = model.predict(X)

print("Learned Weights:", model.weights)
print("Learned Bias:", model.bias)
print("Predicted Labels:", y_pred)
```

```
Learned Weights: [ 0.61526755 -1.03440143]
Learned Bias: 0.8117850524551875
Predicted Labels: [1, 1, 1, 0, 0]
```

Experiment 5: Implement Decision tree classification algorithm

dataset - https://drive.google.com/file/d/1X_o294zDAaUvpifAnKjxBlxLBkMiE3i8/view?usp=drive_link

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
#take train dataset from a drive link
!gdown --id 1X_o294zDAaUvpifAnKjxBlxLBkMiE3i8

data = pd.read_csv('/content/titanic_dataset_train.csv')
```

```
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132:
FutureWarning: Option `--id` was deprecated in version 4.3.1 and will
be removed in 5.0. You don't need to pass it anymore to use a file ID.
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1X_o294zDAaUvpifAnKjxBlxLBkMiE3i8
To: /content/titanic_dataset_train.csv
100% 61.2k/61.2k [00:00<00:00, 62.3MB/s]
```

EDA

```
data.head(2)
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 891,\n  \"fields\":
[\n    {\n      \"column\": \"PassengerId\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 257,\n
\"min\": 1,\n        \"max\": 891,\n        \"num_unique_values\":
891,\n        \"samples\": [\n          710,\n          440,\n
841\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Survived\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Pclass\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 3,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          3,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 891,\n        \"samples\":
[\n          \"Moubarek, Master. Halim Gonios (\\\"William
George\\\")\",\n          \"Kvillner, Mr. Johan Henrik Johannesson\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"female\",\n          \"male\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 14.526497332334042,\
n        \"min\": 0.42,\n        \"max\": 80.0,\n
\"num_unique_values\": 88,\n        \"samples\": [\n          0.75,\n
22.0\n        ],\n        \"semantic_type\": \"\",\n
```

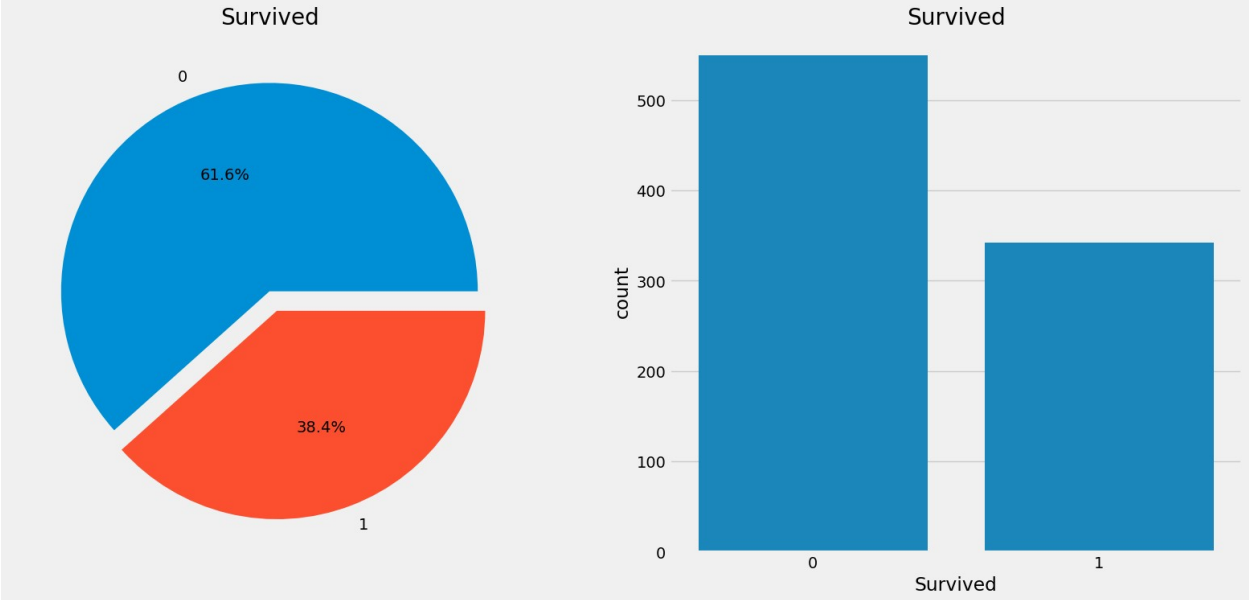
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 0,\n   \"max\": 8,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Parch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 6,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Ticket\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 681,\n   \"samples\": [\n
\"11774\",\n   \"248740\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 49.6934285971809,\n
\"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 248,\n   \"samples\": [\n
11.2417,\n   51.8625\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Cabin\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 147,\n
\"samples\": [\n   \"D45\",\n   \"B49\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"data"}

```
data.isnull().sum()
```

```
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
```

```
f, ax = plt.subplots(1, 2, figsize=(18,8))
data['Survived'].value_counts().plot.pie(explode=[0,0.1],
autopct='%1.1f%%', ax=ax[0])
```

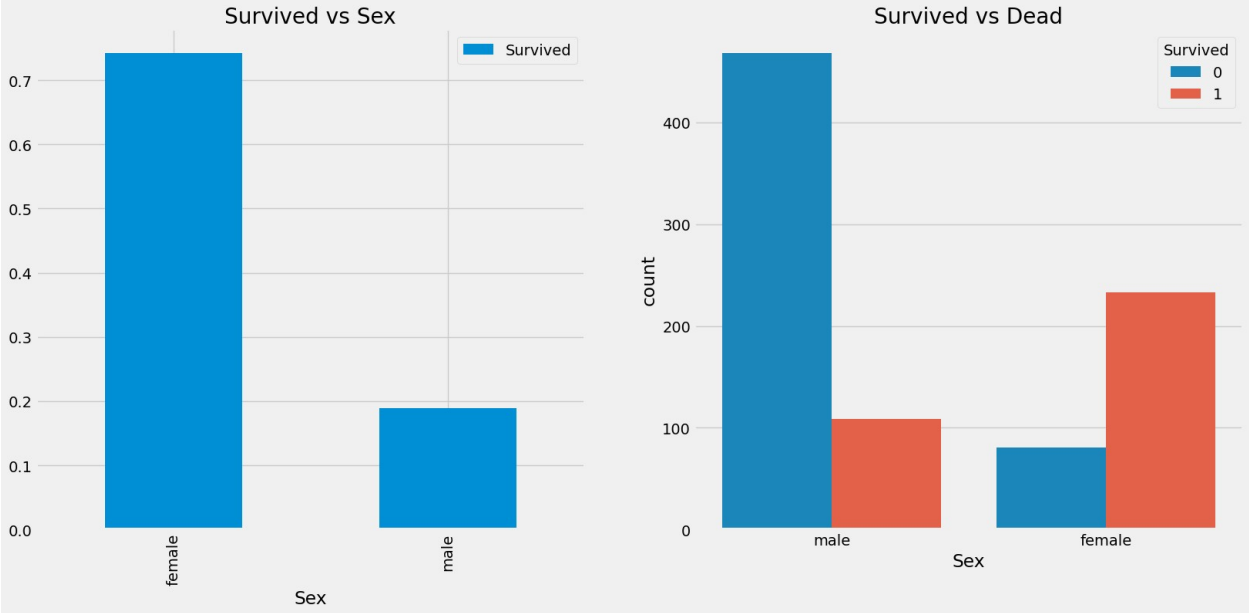
```
ax[0].set_title('Survived')
ax[0].set_ylabel('')
sns.countplot(x='Survived', data=data, ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```




```
data.groupby(['Sex','Survived'])['Survived'].count()
```

```
Sex     Survived
female  0            81
        1           233
male    0           468
        1           109
Name: Survived, dtype: int64
```

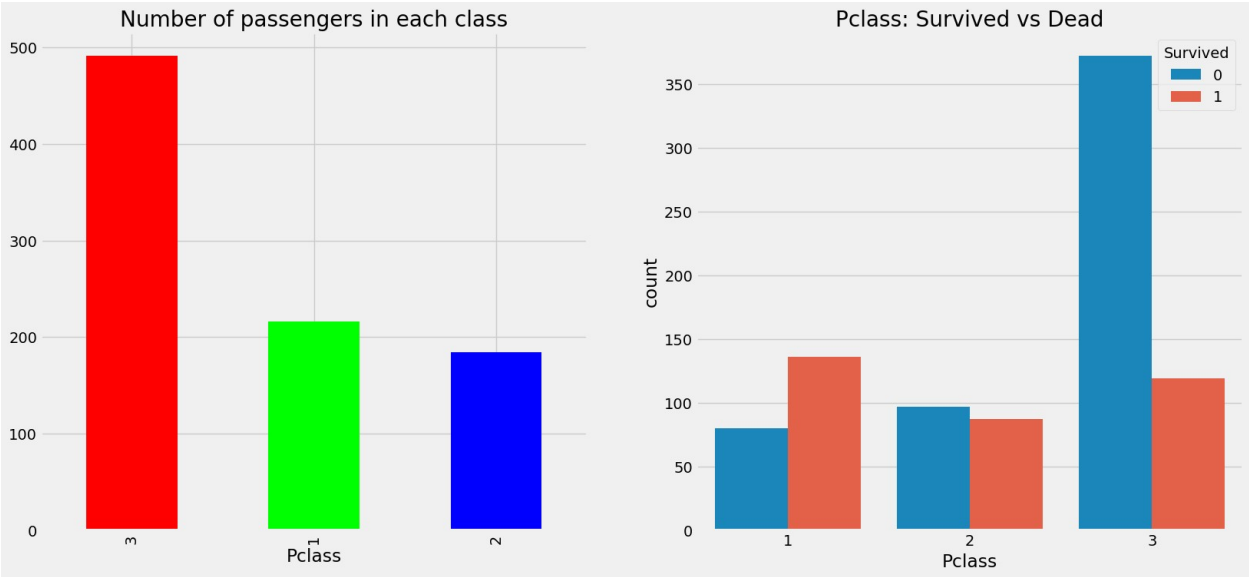
```
f, ax = plt.subplots(1,2,figsize=(18,8))
data[['Sex','Survived']].groupby(['Sex']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survived vs Sex')
sns.countplot(x='Sex',hue='Survived', data=data, ax=ax[1])
ax[1].set_title('Survived vs Dead')
plt.show()
```




```
pd.crosstab(data['Pclass'], data['Survived'],
margins=True).style.background_gradient(cmap='summer_r')
```

```
<pandas.io.formats.style.Styler at 0x7ddde432edd0>
```

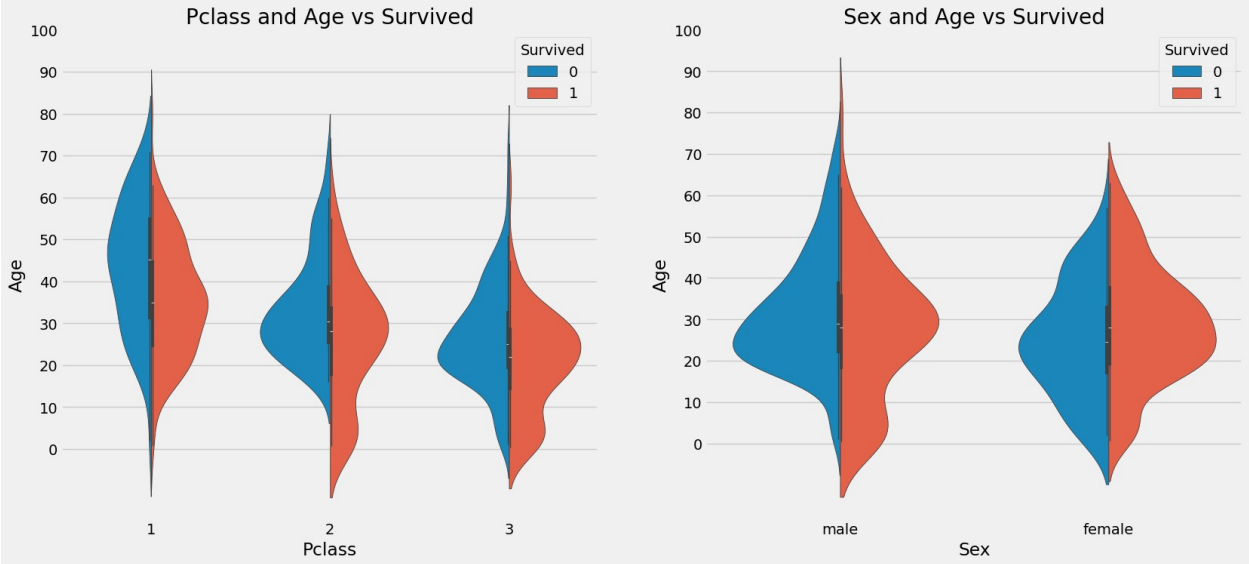
```
f, ax = plt.subplots(1,2,figsize=(18,8))
data['Pclass'].value_counts().plot.bar(color=['#FF0000','#00FF00','#00
00FF'], ax=ax[0])
ax[0].set_title('Number of passengers in each class')
sns.countplot(x=data['Pclass'],hue='Survived', data=data, ax=ax[1])
ax[1].set_title('Pclass: Survived vs Dead')
plt.show()
```



```
print('Oldest Passenger was of:', data['Age'].max(), 'Years')
print('Youngest Passenger was of:', data['Age'].min(), 'Years')
print('Average Age on the ship:', data['Age'].mean(), 'Years')
print('Median Age on the ship:', np.nanmedian(data['Age']), 'Years')
```

```
Oldest Passenger was of: 80.0 Years
Youngest Passenger was of: 0.42 Years
Average Age on the ship: 29.69911764705882 Years
Median Age on the ship: 28.0 Years
```

```
f, ax=plt.subplots(1,2,figsize=(18,8))
sns.violinplot(x='Pclass',y='Age', hue='Survived', data=data,
split=True, ax=ax[0])
ax[0].set_title('Pclass and Age vs Survived')
ax[0].set_yticks(range(0,110,10))
sns.violinplot(x='Sex', y='Age', hue='Survived',data=data, split=True,
ax=ax[1])
ax[1].set_title('Sex and Age vs Survived')
ax[1].set_yticks(range(0,110,10))
plt.show()
```




```
data['Initial']=0
for i in data:
  data['Initial']=data.Name.str.extract('([A-Za-z]+)\.')

pd.crosstab(data['Initial'],data['Sex']).T.style.background_gradient(c
map='summer_r')
```

```
<pandas.io.formats.style.Styler at 0x7ddd9c5bc070>
```

```
data.groupby('Initial')['Age'].mean()
```

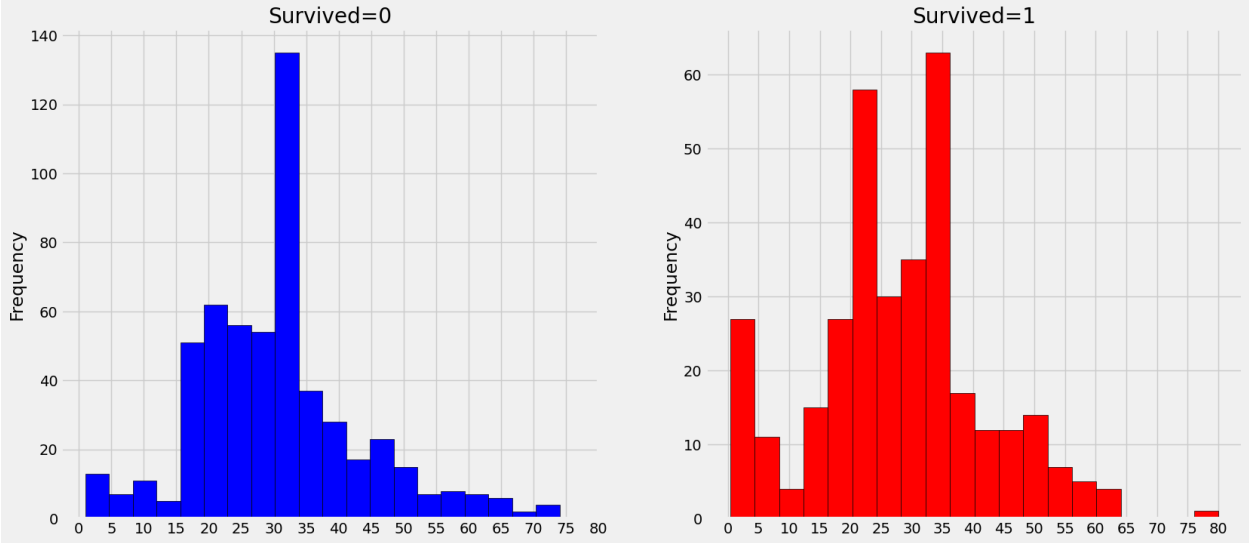
```
Initial
Capt        70.000000
Col         58.000000
Countess    33.000000
Don         40.000000
Dr          42.000000
Jonkheer    38.000000
Lady        48.000000
Major       48.500000
Master       4.574167
Miss        21.773973
Mlle        24.000000
Mme         24.000000
Mr          32.368090
Mrs         35.898148
Ms          28.000000
Rev         43.166667
Sir         49.000000
Name: Age, dtype: float64
```

```
#handling null data in ages
data.loc[(data.Age.isnull())&(data.Initial=='Mr'), 'Age']=33
data.loc[(data.Age.isnull())&(data.Initial=='Mrs'), 'Age']=36
data.loc[(data.Age.isnull())&(data.Initial=='Master'), 'Age']=5
data.loc[(data.Age.isnull())&(data.Initial=='Miss'),'Age']=22
data.loc[(data.Age.isnull())&(data.Initial=='Don'),'Age']=40
data.loc[(data.Age.isnull())&(data.Initial=='Rev'),'Age']=43
data.loc[(data.Age.isnull())&(data.Initial=='Other'),'Age'] =46
```

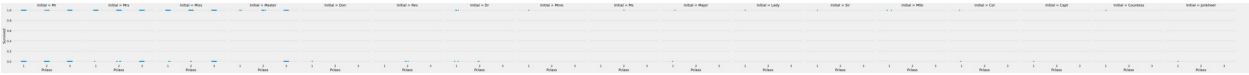
```
data['Age'].isnull().any()
```

```
True
```

```
f, ax=plt.subplots(1,2,figsize=(18,8))
data[data['Survived']==0].Age.plot.hist(ax=ax[0], bins=20,
edgecolor='black', color="#0000ff")
ax[0].set_title('Survived=0')
x1=list(range(0,85,5))
ax[0].set_xticks(x1)
data[data['Survived']==1].Age.plot.hist(ax=ax[1],
bins=20,edgecolor='black', color="#ff0000")
ax[1].set_title('Survived=1')
x2=list(range(0,85,5))
ax[1].set_xticks(x2)
plt.show()
```



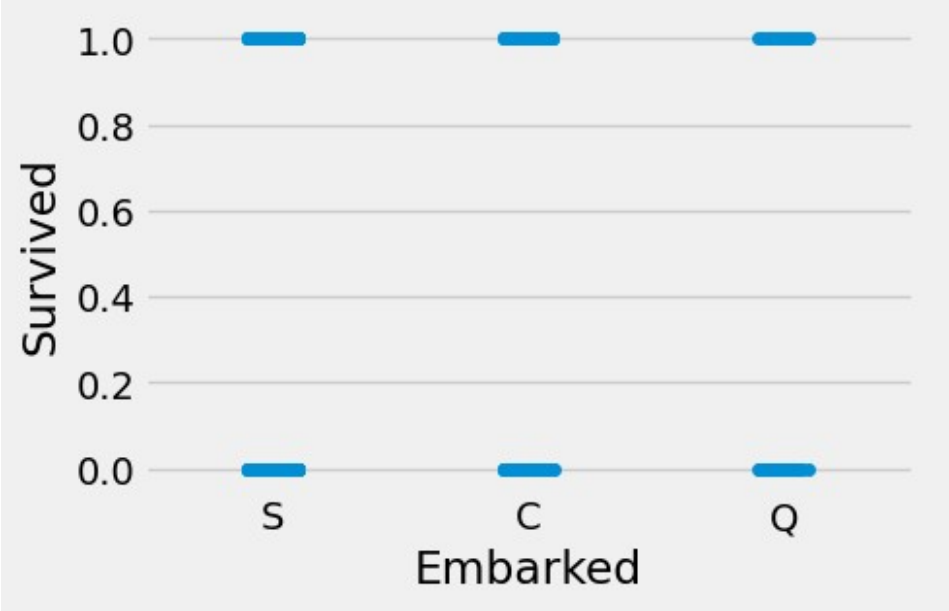

```
sns.catplot(x='Pclass', y='Survived', col='Initial',data=data)
plt.show()
```



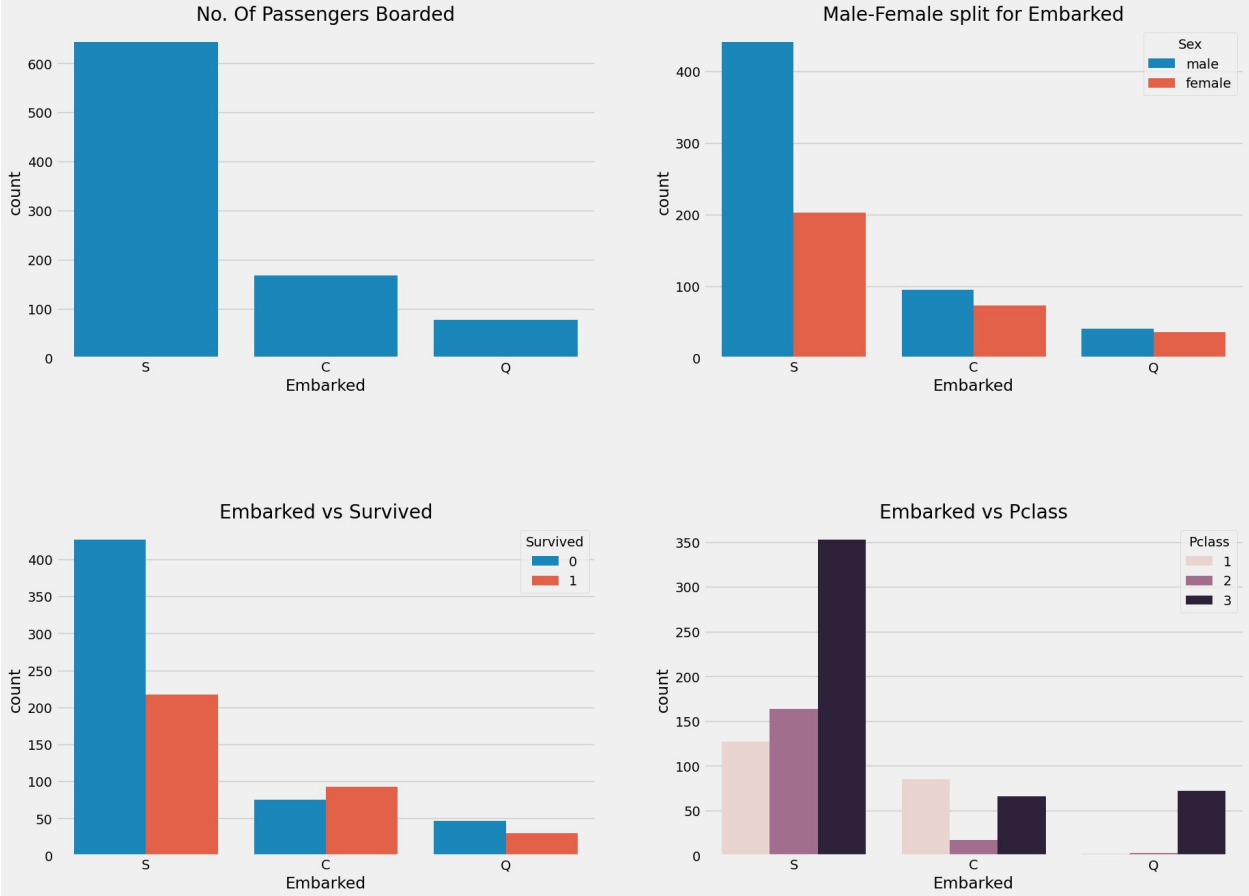
```
# attr embarked - where did people board from
pd.crosstab([data.Embarked,data.Pclass],
[data.Sex,data.Survived],margins=True).style.background_gradient(cmap=
'summer_r')
```

```
<pandas.io.formats.style.Styler at 0x7ddd9c331bd0>
```

```
sns.catplot(x='Embarked', y='Survived', data=data)
fig=plt.gcf()
fig.set_size_inches(5,3)
plt.show()
```



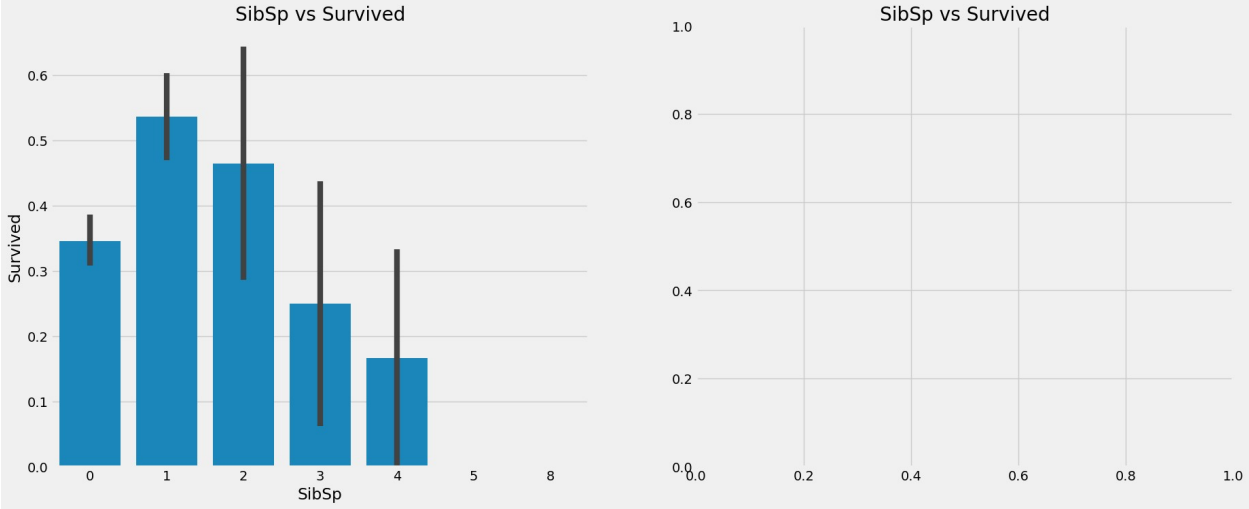

```
f, ax=plt.subplots(2, 2, figsize=(20,15))
sns.countplot(x='Embarked', data=data, ax=ax[0,0])
ax[0,0].set_title('No. Of Passengers Boarded')
sns.countplot(x='Embarked', hue='Sex', data=data, ax=ax[0,1])
ax[0,1].set_title('Male-Female split for Embarked')
sns.countplot(x='Embarked', hue='Survived', data=data, ax=ax[1,0])
ax[1,0].set_title('Embarked vs Survived')
sns.countplot(x='Embarked', hue='Pclass', data=data, ax=ax[1,1])
ax[1,1].set_title('Embarked vs Pclass')
plt.subplots_adjust(wspace=0.2, hspace=0.5 )
plt.show()
```

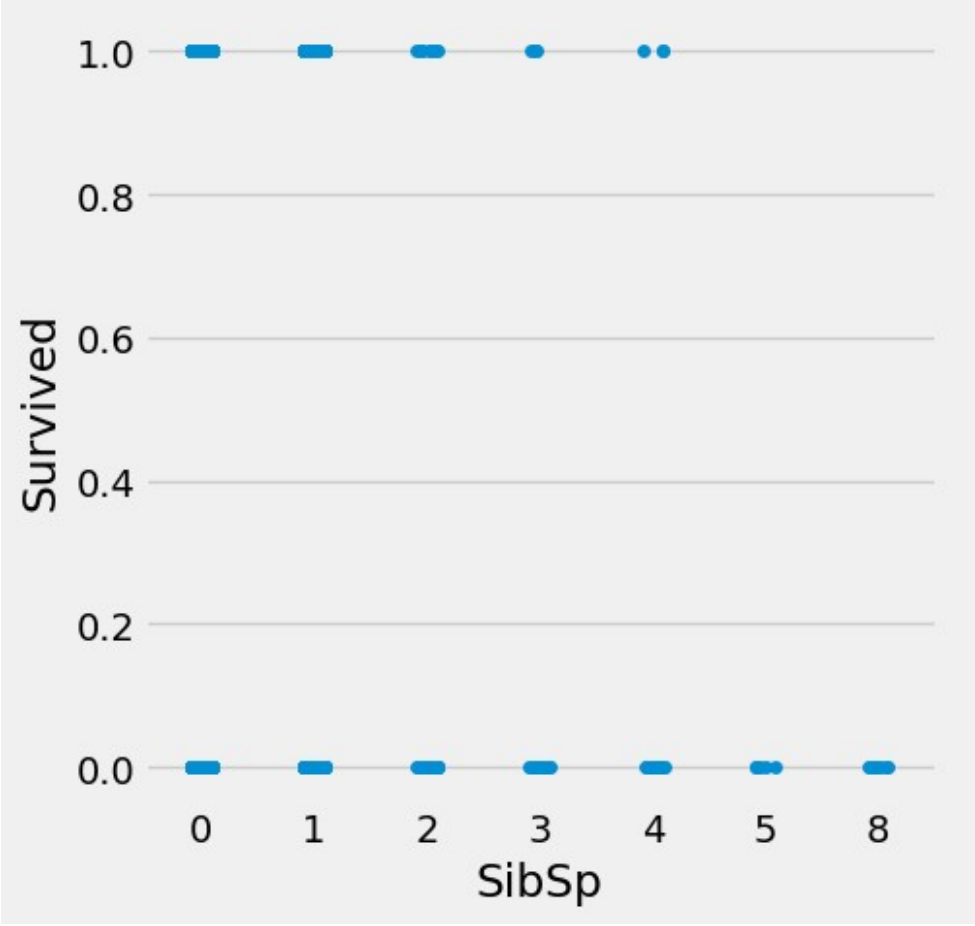



```
#attr sibsp - no. of sibling or spoused onboard
pd.crosstab([data.SibSp],data.Survived).style.background_gradient(cmap
='summer_r')
```

```
<pandas.io.formats.style.Styler at 0x7ddd9c4e5840>
```

```
f, ax= plt.subplots(1,2, figsize=(20,8))
sns.barplot(x='SibSp', y='Survived',data=data,ax=ax[0])
ax[0].set_title('SibSp vs Survived')
sns.catplot(x='SibSp', y='Survived',data=data,ax=ax[1])
ax[1].set_title('SibSp vs Survived')
plt.show()
```





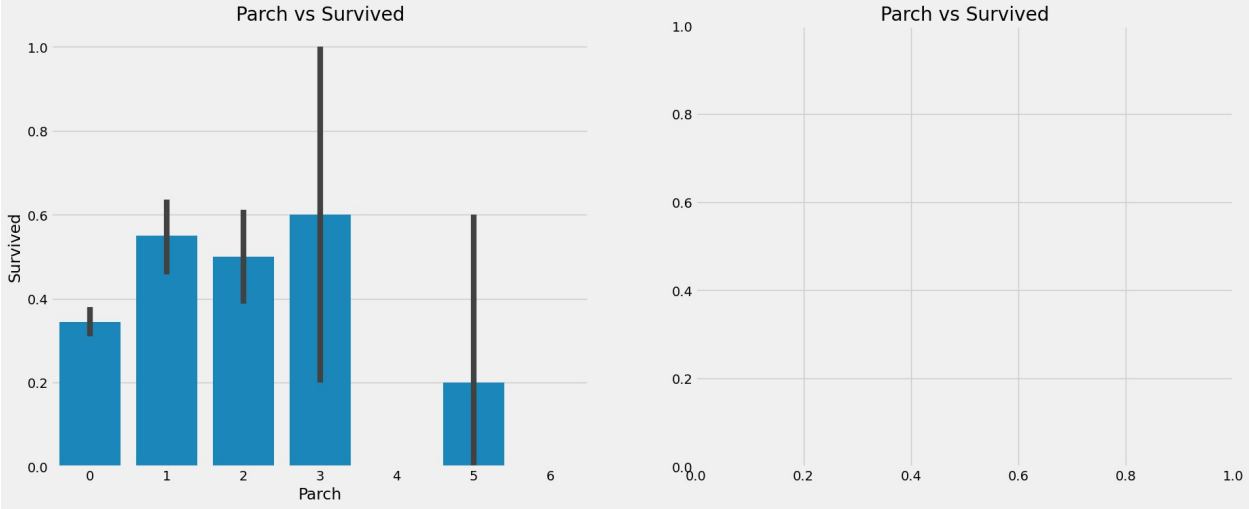

```
pd.crosstab(data.SibSp,
data.Pclass).style.background_gradient(cmap='summer_r')
```

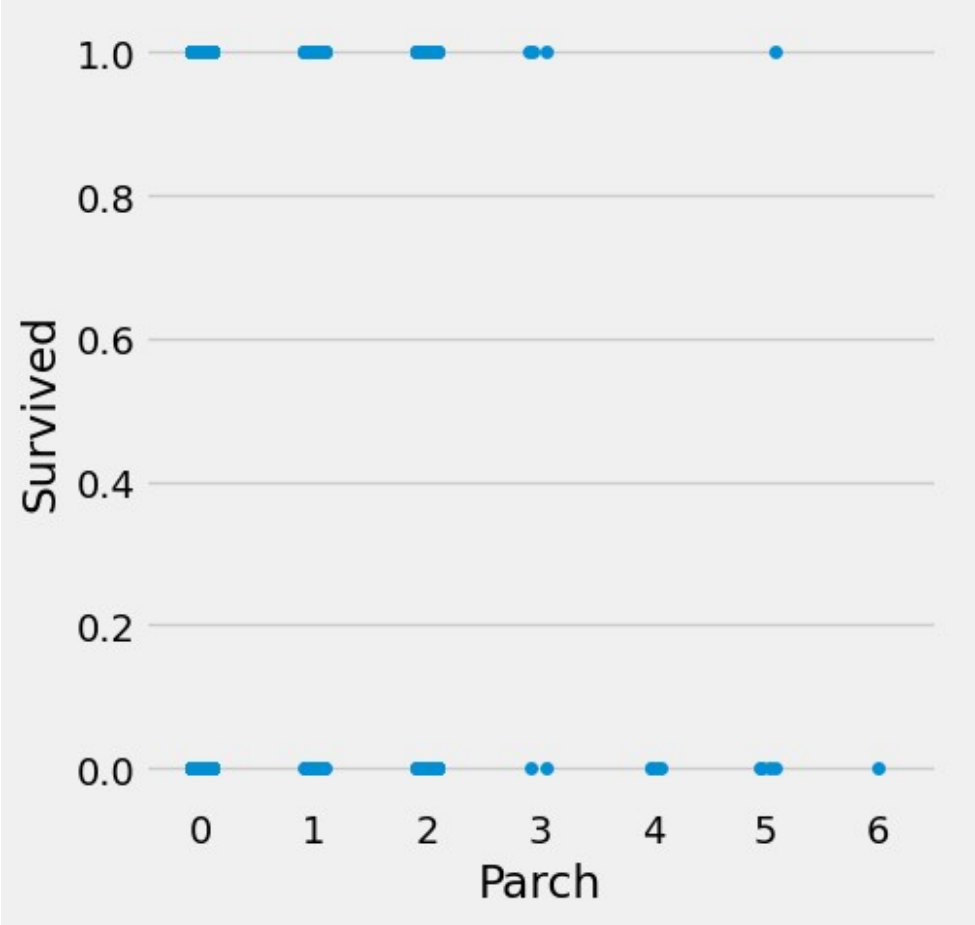
```
<pandas.io.formats.style.Styler at 0x7ddd96871840>
```

```
#attr parch - no. of parents or children onboard
pd.crosstab(data.Parch,
data.Pclass).style.background_gradient(cmap='summer_r')
```

```
<pandas.io.formats.style.Styler at 0x7ddd9cc25ab0>
```

```
f, ax=plt.subplots(1, 2, figsize=(20,8))
sns.barplot(x='Parch', y='Survived', data=data, ax=ax[0])
ax[0].set_title('Parch vs Survived')
sns.catplot(x='Parch', y='Survived', data=data, ax=ax[1])
ax[1].set_title('Parch vs Survived')
plt.show()
```

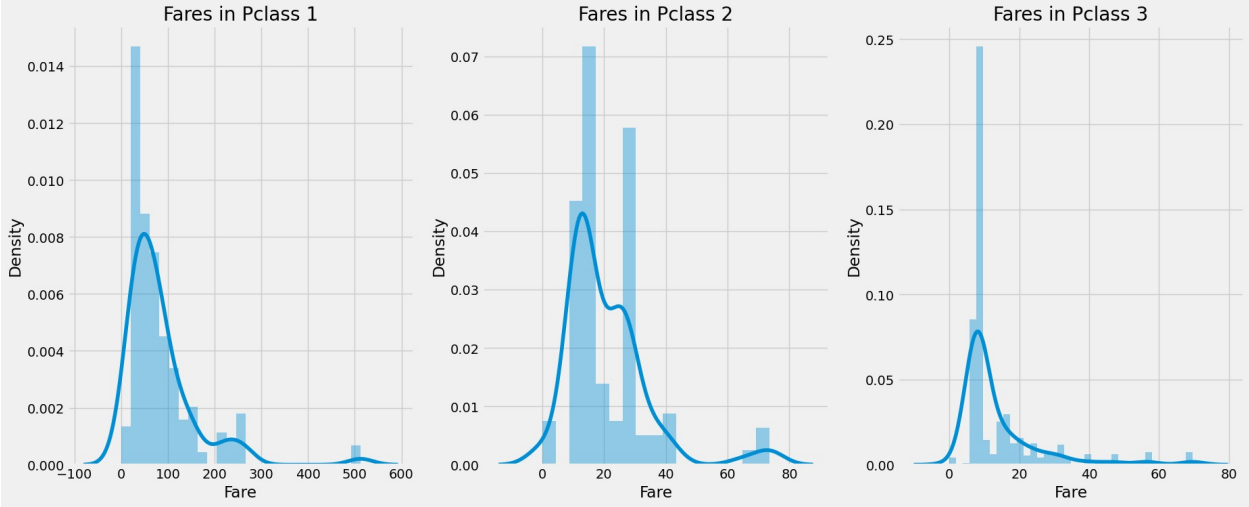




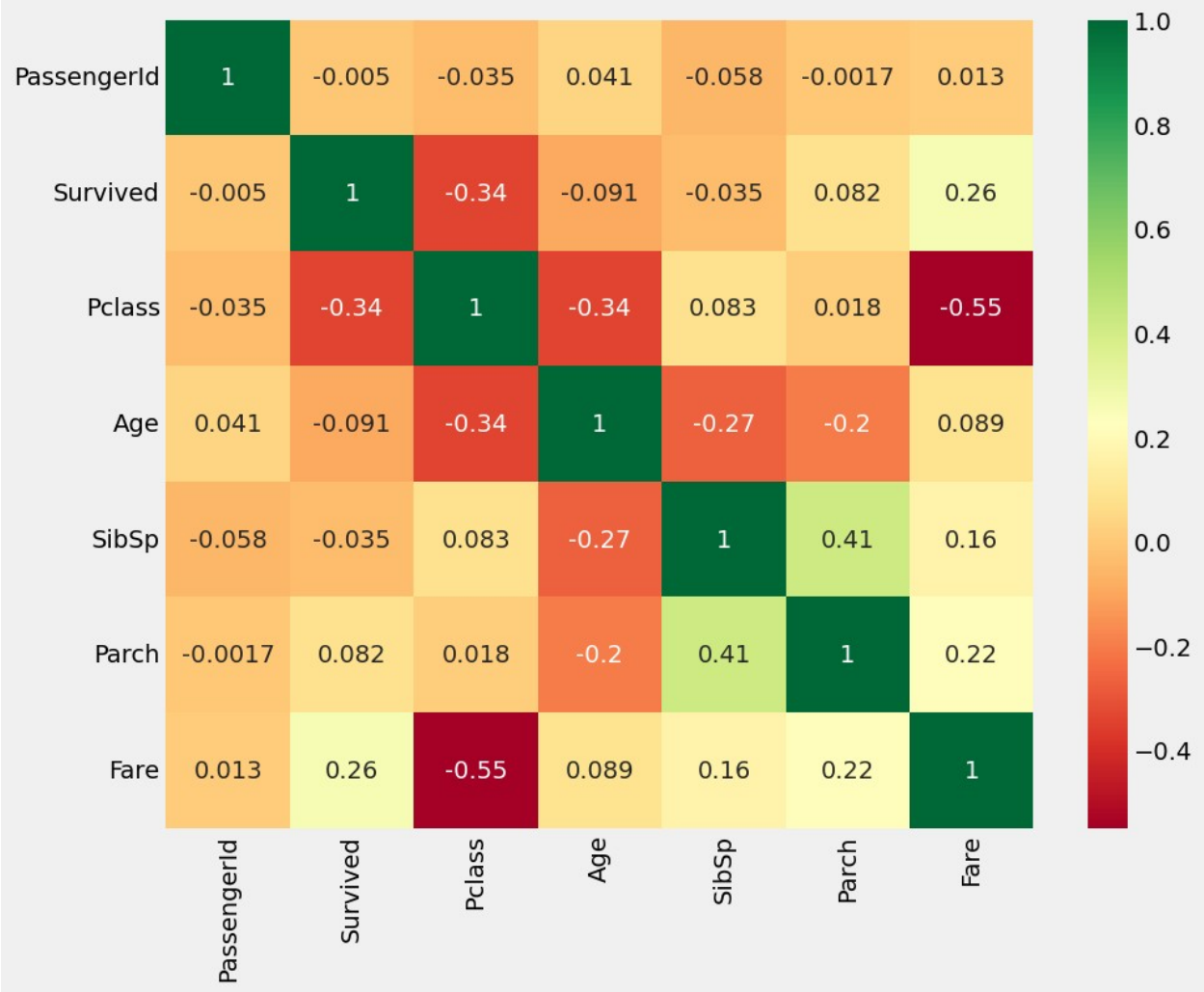

```
print('Highest Fare was:', data['Fare'].max())
print('Lowest Fare was:',data['Fare'].min())
print('Average Fare was:', data['Fare'].mean())
```

```
Highest Fare was: 512.3292
Lowest Fare was: 0.0
Average Fare was: 32.204207968574636
```

```
f,ax=plt.subplots(1,3, figsize=(20,8))
sns.distplot(data[data['Pclass']==1].Fare,ax=ax[0])
ax[0].set_title('Fares in Pclass 1')
sns.distplot(data[data['Pclass']==2].Fare,ax=ax[1])
ax[1].set_title('Fares in Pclass 2')
sns.distplot(data[data['Pclass']==3].Fare ,ax=ax[2])
ax[2].set_title('Fares in Pclass 3')
plt.show()
```

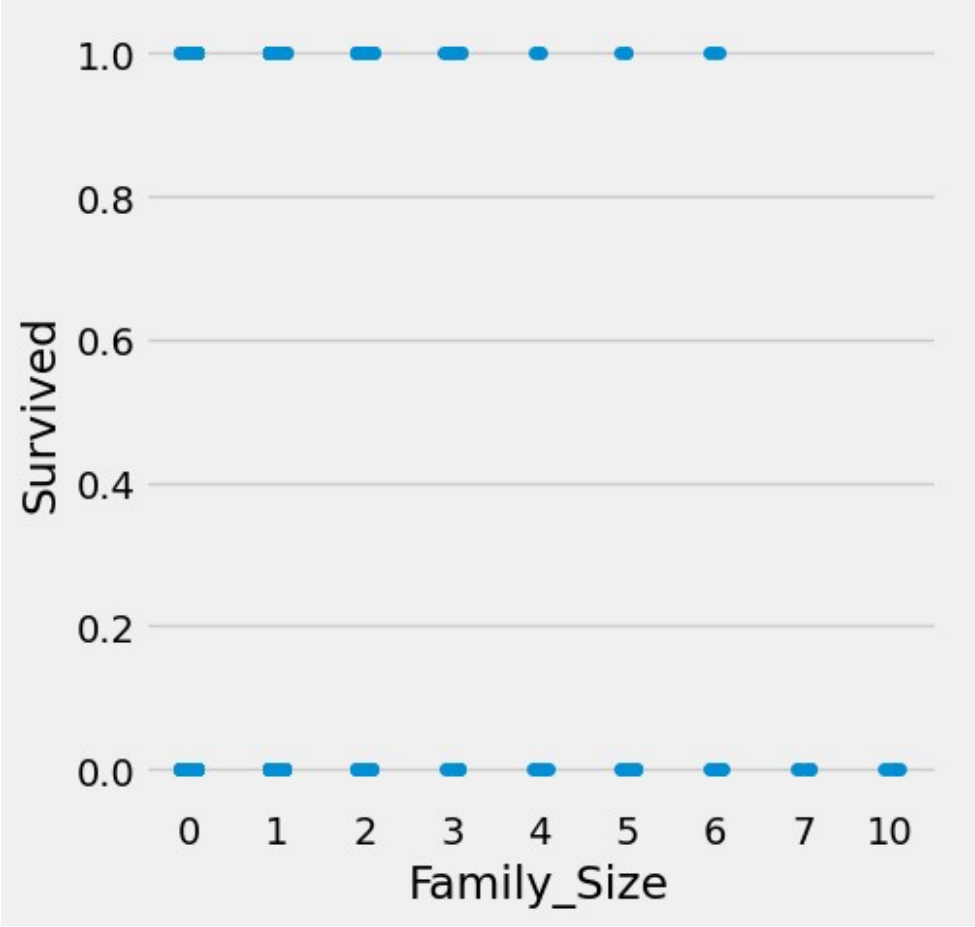


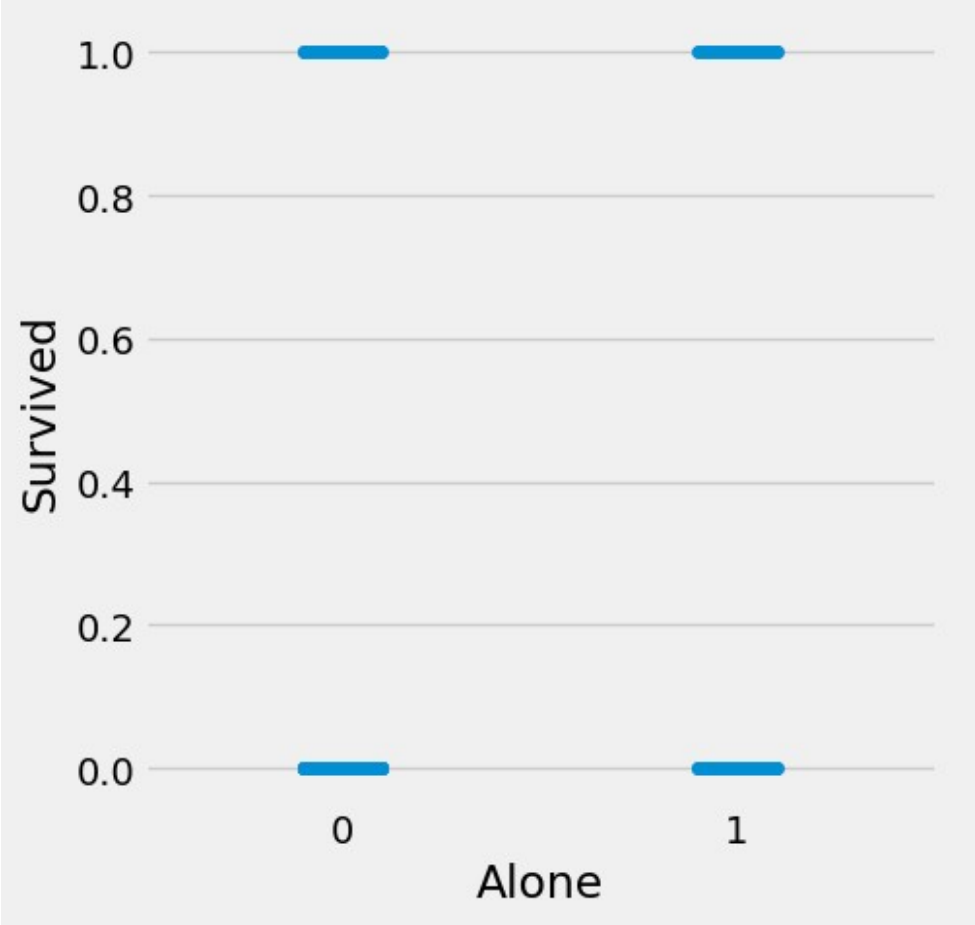

```
#correlation between features
sns.heatmap(data.select_dtypes(include=['number']).corr(),annot=True,
cmap='RdYlGn')
fig=plt.gcf()
fig.set_size_inches(10,8)
plt.show()
```



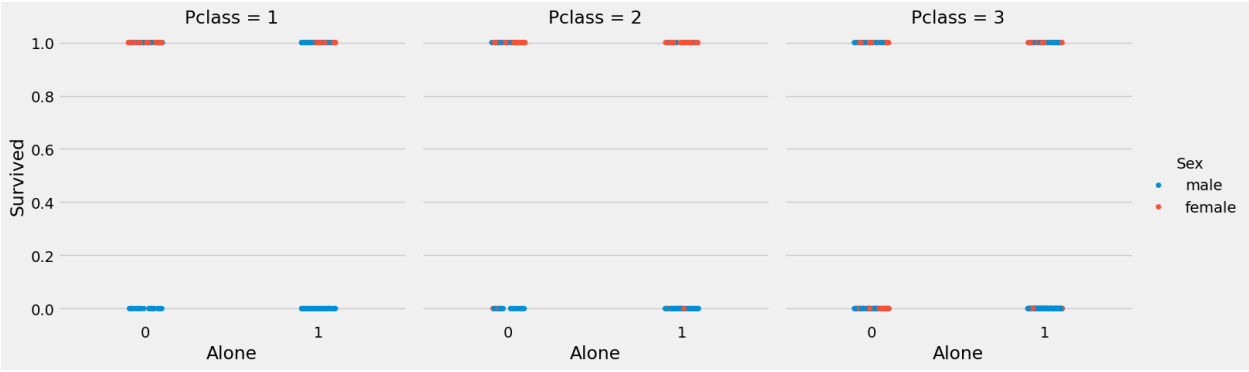

```
# data engineering & cleaning
data['Family_Size']=0
data['Family_Size']=data['Parch']+data['SibSp']
data['Alone']=0
data.loc[data.Family_Size==0,'Alone']=1

f,ax=plt.subplots(1,2,figsize=(18,6))
sns.catplot(x='Family_Size', y='Survived', data=data, ax=ax[0])
ax[0].set_title('Family_Size vs Survived')
sns.catplot(x='Alone',y='Survived', data=data, ax=ax[1])
ax[1].set_title('Alone vs Survived')
plt.show()
```


Family_Size vs Survived
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Alone vs Survived
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Survived
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
10
Family_Size



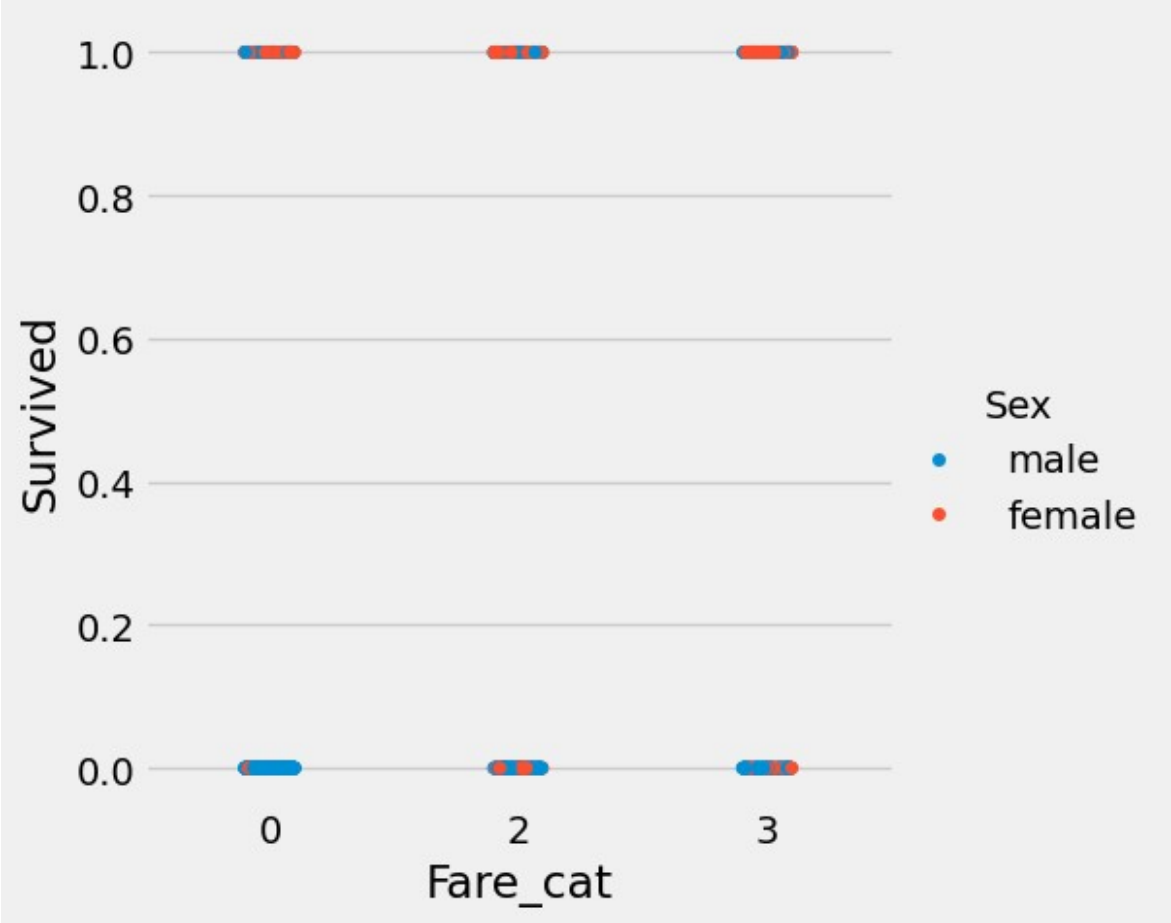

```
sns.catplot(x='Alone', y='Survived', data=data, hue='Sex',
col='Pclass')
plt.show()
```




```
data['Fare_Range']=pd.qcut(data['Fare'],4)
data.groupby(['Fare_Range'])
['Survived'].mean().to_frame().style.background_gradient(cmap='summer_
r')
```

```
<pandas.io.formats.style.Styler at 0x7ddd9c73afe0>
```

```
data['Fare_cat']=0
data.loc[data['Fare']<=7.91, 'Fare_cat']=0
data.loc[(data['Fare']<=14.454)&(data['Fare']>14.454),'Fare_cat']=1
data.loc[(data['Fare']<=31)&(data['Fare']>14.454),'Fare_cat']=2
data.loc[(data['Fare']<=513)&(data['Fare']>31),'Fare_cat']=3

sns.catplot(x='Fare_cat', y='Survived', data=data,hue='Sex')
plt.show()
```




```
data['Sex'].replace(['male','female'],[0,1], inplace=True)
data['Embarked'].replace(['S','Q', 'C'], [0,1,2], inplace=True)
data['Initial'] = data['Initial'].factorize()[0]


for x in data.columns:
  print(f'{x}:{data[x].dtype}')
```

```
PassengerId:int64
Survived:int64
Pclass:int64
Name:object
```

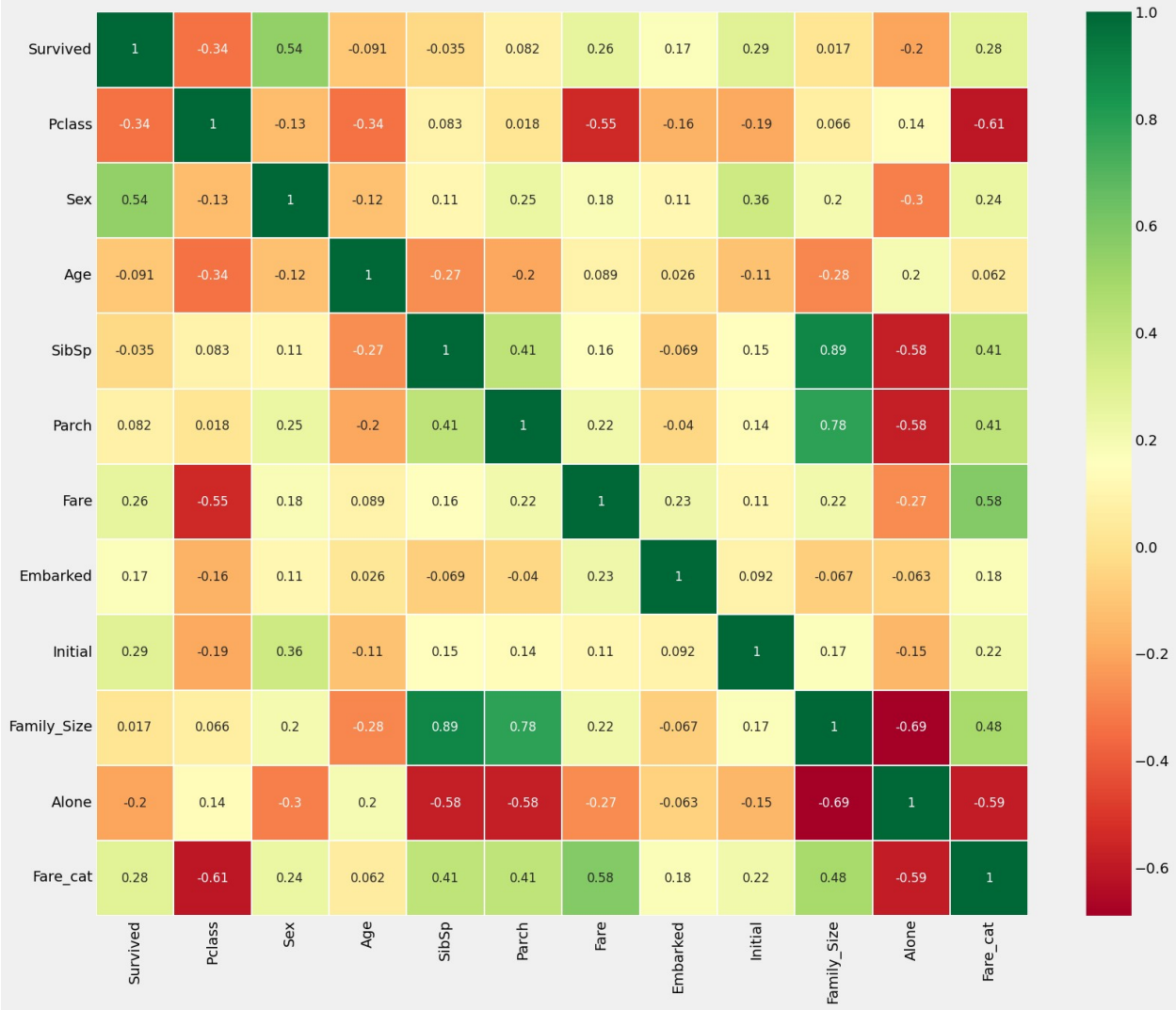
```
Sex:int64
Age:float64
SibSp:int64
Parch:int64
Ticket:object
Fare:float64
Cabin:object
Embarked:float64
Initial:int64
Family_Size:int64
Alone:int64
Fare_Range:category
Fare_cat:int64

data =
data.drop(columns=['Name','PassengerId','Ticket','Fare_Range','Cabin']
, axis=1)

---------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
<ipython-input-50-69bd68845abd> in <cell line: 1>()
----> 1 data =
data.drop(columns=['Name','PassengerId','Ticket','Fare_Range','Cabin']
, axis=1)

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in
drop(self, labels, axis, index, columns, level, inplace, errors)
   5342                 weight  1.0     0.8
   5343         """
-> 5344         return super().drop(
   5345             labels=labels,
   5346             axis=axis,

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in
drop(self, labels, axis, index, columns, level, inplace, errors)
   4709         for axis, labels in axes.items():
   4710             if labels is not None:
-> 4711                 obj = obj._drop_axis(labels, axis,
level=level, errors=errors)
   4712
   4713         if inplace:

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in
_drop_axis(self, labels, axis, level, errors, only_slice)
   4751                 new_axis = axis.drop(labels, level=level,
errors=errors)
   4752             else:
-> 4753                 new_axis = axis.drop(labels, errors=errors)
```

```
   4754             indexer = axis.get_indexer(new_axis)
   4755

/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in
drop(self, labels, errors)
   6998         if mask.any():
   6999             if errors != "ignore":
-> 7000                 raise KeyError(f"{labels[mask].tolist()} not
found in axis")
   7001             indexer = indexer[~mask]
   7002         return self.delete(indexer)

KeyError: "['Name', 'PassengerId', 'Ticket', 'Fare_Range', 'Cabin']
not found in axis"
```

```
for x in data.columns:
  print(f'{x}:{data[x].dtype}')
```

```
Survived:int64
Pclass:int64
Sex:int64
Age:float64
SibSp:int64
Parch:int64
Fare:float64
Embarked:float64
Initial:int64
Family_Size:int64
Alone:int64
Fare_cat:int64
```

```
sns.heatmap(data.corr(), annot=True, cmap='RdYlGn', linewidths=0.2,
annot_kws={'size': 12})
fig=plt.gcf()
fig.set_size_inches(18,15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```




```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn import metrics

data['Age'].fillna(data['Age'].mean(), inplace=True)

from sklearn.model_selection import train_test_split

# Split data into train and test sets
train, test = train_test_split(data, test_size=0.3, random_state=0,
stratify=data['Survived'])

# Separate features and labels for train and test sets
train_X = train.drop('Survived', axis=1)
train_Y = train['Survived']
test_X = test.drop('Survived', axis=1)
test_Y = test['Survived']
```

```
# Features and labels for the entire dataset
X = data.drop('Survived', axis=1)
Y = data['Survived']

# Check for missing values
missing_val_count_by_column = (data.isnull().sum())
print(missing_val_count_by_column)
```

```
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       2
Initial        0
Family_Size    0
Alone          0
Fare_cat       0
dtype: int64
```

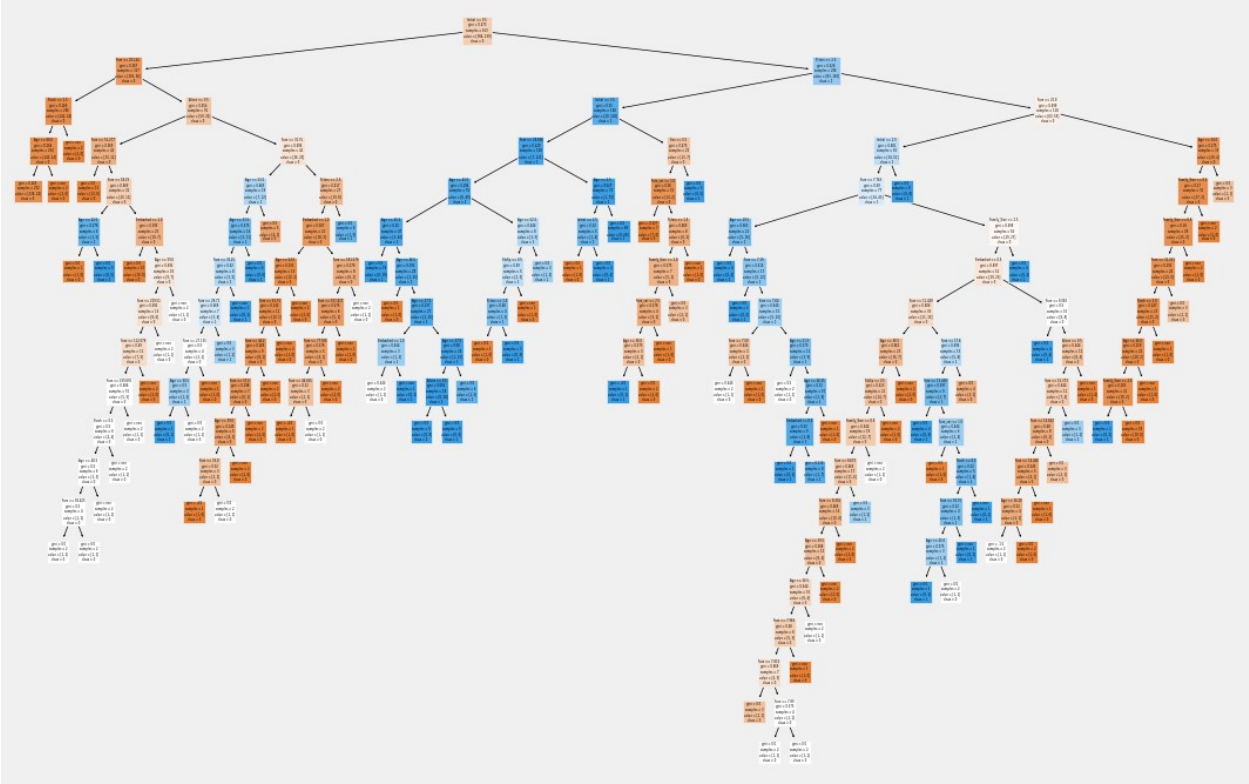
```
model=DecisionTreeClassifier(criterion='gini')
model.fit(train_X, train_Y)
prediction=model.predict(test_X)

print('Accuracy for Decision Tree is',
metrics.accuracy_score(prediction, test_Y))
```

```
Accuracy for Decision Tree is 0.8208955223880597
```

```
feature_names = train_X.columns
feature_names
```

```
Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked',
'Initial',
       'Family_Size', 'Alone', 'Fare_cat'],
      dtype='object')
```

```
plt.figure(figsize=(12, 8))
plot_tree(model, filled=True, feature_names=feature_names,
class_names=['0','1'])
plt.show()
```



Experiment 6 - Implement Support Vector Machines (SVM) for non-linear classification

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

cancer = datasets.load_breast_cancer()
X = cancer.data
y = cancer.target
print(X)
print("\n")
print(y)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1
0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0
1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1
0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0
1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0
1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1
1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0
1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1
0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1
1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0
1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1
0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1]

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

svm_classifier = SVC(kernel = 'linear', C = 1.0)
svm_rbf = SVC(kernel='rbf', C=1.0)  # Initialize with RBF kernel

svm_classifier.fit(X_train, y_train)
svm_rbf.fit(X_train, y_train)  # Train the model

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.96

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.91      0.94        43
           1       0.95      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114


Confusion Matrix:
 [[39  4]
 [ 1 70]]
```

Experiment 7 - Implement ensemble methods to combine different models

```python
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import VotingClassifier

# Load the dataset as an example
data = load_breast_cancer()
X, y = data.data, data.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# BAGGING (BootStrap Aggregating)
# Bagging involves training multiple instances of the same model on
different subsets of the training data and averaging their
# predictions. This helps reduce overfitting and improve model
stability. The most common example is the Random Forest algorithm.
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)
rf_classifier.fit(X_train, y_train)
rf_predictions = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Predictions:", rf_predictions)
```

```
Predictions: [1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0
1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1
1 0
 1 1 0]
```

```
# BOOSTING
# Boosting focuses on improving the performance of a weak learner
# (e.g., a decision tree) by giving more weight to the data points
# that were misclassified in previous iterations. Popular boosting
# algorithms include AdaBoost and Gradient Boosting.
# AdaBoost
adaboost_classifier = AdaBoostClassifier(n_estimators=100,
random_state=42)
adaboost_classifier.fit(X_train, y_train)
adaboost_predictions = adaboost_classifier.predict(X_test)
adaboost_accuracy = accuracy_score(y_test, adaboost_predictions)
print("Predictions:", adaboost_predictions)
print(f"Accuracy: {adaboost_accuracy}\n")
# Gradient Boost
gradientboost_classifier =
GradientBoostingClassifier(n_estimators=100, random_state=42)
gradientboost_classifier.fit(X_train, y_train)
gradientboost_predictions = gradientboost_classifier.predict(X_test)
gradientboost_accuracy = accuracy_score(y_test,
gradientboost_predictions)
print("Predictions:", gradientboost_predictions)
print(f"Accuracy: {gradientboost_accuracy}\n")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/
_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the
default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(

Predictions: [1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0
1 0
 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1
1 0
 1 1 0]
Accuracy: 0.9736842105263158

Predictions: [1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0
1 0
```

```
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0
1 0
 1 1 0]
Accuracy: 0.956140350877193


# STACKING
# Stacking involves training multiple diverse models and then
# training another model (meta-learner) to combine their predictions.
# It combines the strengths of different models and can often lead to
# improved performance.
# Create base models
base_models = [
('RandomForest', RandomForestClassifier(n_estimators=100,
random_state=42)),
('AdaBoost', AdaBoostClassifier(n_estimators=100,
random_state=42)),
('GradientBoosting',
GradientBoostingClassifier(n_estimators=100, random_state=42))
]
# Create a meta-classifier (Logistic Regression)
meta_classifier = LogisticRegression()
# Create the stacking ensemble
stacking_classifier = VotingClassifier(estimators=base_models,
voting='soft', n_jobs=-1)
stacking_classifier.fit(X_train, y_train)
stacking_predictions = stacking_classifier.predict(X_test)
stacking_accuracy = accuracy_score(y_test, stacking_predictions)
print("Predictions:", stacking_predictions)
print(f"Stacking Accuracy: {stacking_accuracy}")

Predictions: [1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0
1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0
1 0
 1 1 0]
Stacking Accuracy: 0.956140350877193
```

Experiment 8 - Implement Principal Component Analysis (PCA) technique

```
# PCA using inbuilt methods
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```python
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt

data = load_breast_cancer().data

# Step 1: Standardize the data (mean=0, variance=1)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
# print(scaled_data)
# print("\n")
# Step 2: Perform PCA
n_components = 2 # Number of principal components to retain
pca = PCA(n_components=n_components)
pca.fit(scaled_data)
# Step 3: Transform the data to the new feature space
pca_result = pca.transform(scaled_data)
# Step 4: Explained Variance Ratio
explained_variance_ratio = pca.explained_variance_ratio_
print("Explained Variance Ratios:", explained_variance_ratio)
print("\nTotal Variance Explained:", sum(explained_variance_ratio))
# The transformed data contains the first 'n_components' principal
components
print("\nPCA Result:")
print(pca_result)
```

```
Explained Variance Ratios: [0.44272026 0.18971182]

Total Variance Explained: 0.6324320765155944

PCA Result:
[[ 9.19283683  1.94858307]
 [ 2.3878018  -3.76817174]
 [ 5.73389628 -1.0751738 ]
 ...
 [ 1.25617928 -1.90229671]
 [10.37479406  1.67201011]
 [-5.4752433  -0.67063679]]
```

Case Study: "A Survey of Reinforcement Learning in Autonomous Robotics: Challenges, Trends, and Applications"

Summary:

The research paper titled "A Survey of Reinforcement Learning in Autonomous Robotics" provides a comprehensive examination of the utilization of reinforcement learning (RL) techniques within the domain of autonomous robotics. Reinforcement learning, a subset of machine learning, has garnered substantial attention due to its potential to enable robots to autonomously acquire complex skills and adapt to dynamic environments. This summary will delve into the paper's primary components, offering a more detailed understanding of its content.

Introduction and Background:

The paper initiates by establishing the fundamental concepts of reinforcement learning, laying the groundwork for readers unfamiliar with the topic. It explains key RL concepts such as Markov decision processes, reward functions, and the trade-off between exploration and exploitation. This introductory section sets the stage for the subsequent exploration of RL in robotics.

Challenges in RL for Robotics:

One of the paper's core focuses is an in-depth analysis of the challenges inherent in applying RL to robotics. Safety is identified as a paramount concern, as robots must operate in environments that may pose risks to themselves and others. Sample efficiency is another challenge, particularly important in real-world scenarios where collecting data can be costly and time-consuming. Additionally, the need for real-time decision-making is emphasized, as robots must make rapid and precise choices to navigate their surroundings effectively.

Solutions and Techniques:

To address these challenges, the paper discusses various techniques and strategies employed by researchers in the field of RL for robotics. Simulation, for example, is explored as a means to train RL agents in safe and efficient environments before deploying them in the real world. Domain randomization is presented as a technique to make trained agents more adaptable to diverse environments. Safe exploration strategies are highlighted as essential for preventing catastrophic failures during the learning process.

Trends in RL for Robotics:

The paper brings attention to the latest trends in RL for robotics. Deep reinforcement learning (DRL) is a prominent trend, illustrating how deep neural networks are integrated into RL algorithms to handle high-dimensional sensor data and complex control tasks. It also explores the fusion of sensor data, perception systems, and control algorithms with RL, emphasizing the role of sensorimotor skills in robotic tasks. The development of custom hardware and software platforms tailored to RL-based robotics applications is discussed as a key trend, furthering the integration of RL into practical robotic systems.

Applications in Real-World Scenarios:

The authors showcase numerous applications where RL is proving to be a game-changer in the field of robotics. These applications span a wide range, including autonomous navigation, robotic manipulation, drone control, and autonomous vehicles. By providing real-world examples and case studies, the paper illustrates how RL is enabling robots to tackle complex tasks and adapt to dynamic environments successfully.

In conclusion, "A Survey of Reinforcement Learning in Autonomous Robotics" offers a comprehensive exploration of the role of RL in shaping the future of autonomous robotic systems. By addressing challenges, highlighting trends, and presenting real-world applications, this paper serves as an invaluable resource for researchers, engineers, and robotics enthusiasts seeking to understand the integration of RL techniques in the field of robotics and its potential.

Arnav Malvia
2103109
C32
ML
Group: 44

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Assignment :- 1

Q1 What is learning? Explain different types of learning with example

Ans 1 Machine learning is a category of Artificial Intelligence. In machine learning computers have the ability to learn themselves, explicit programming is not required. Machine learning focuses on the study and development of algorithms that can learn from data and also make predictions on data.

Machine learning mainly focuses on the design and development of computer programs that can teach themselves to grow and change when exposed to new data. Using machine learning we can collect information from a dataset by asking the computer to make some sense from data. Machine learning is turning data into information.

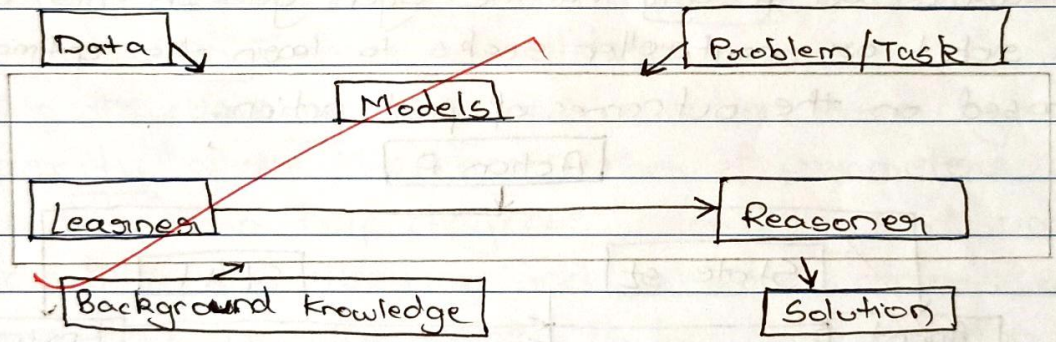
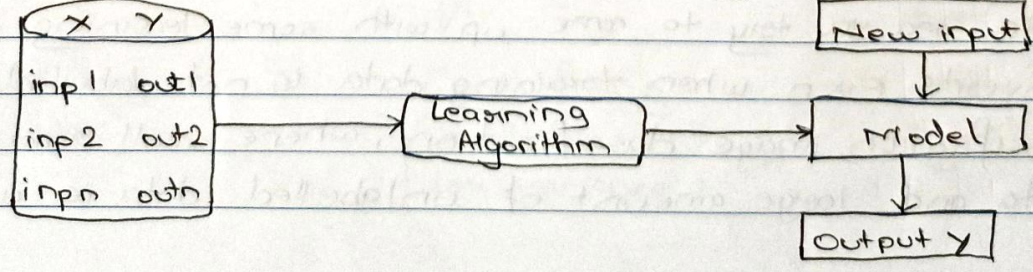


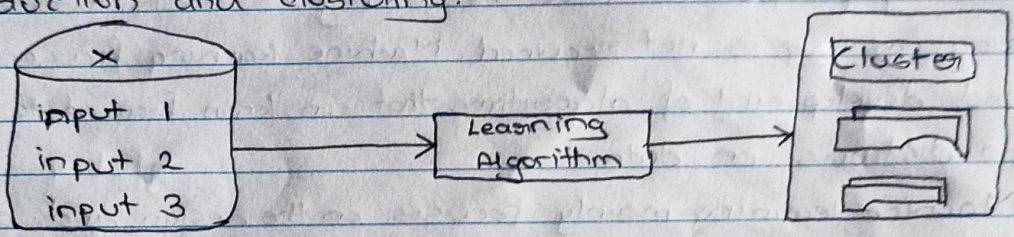
Fig: Schematic Diagram of Machine Learning

Types of Machine Learning:-

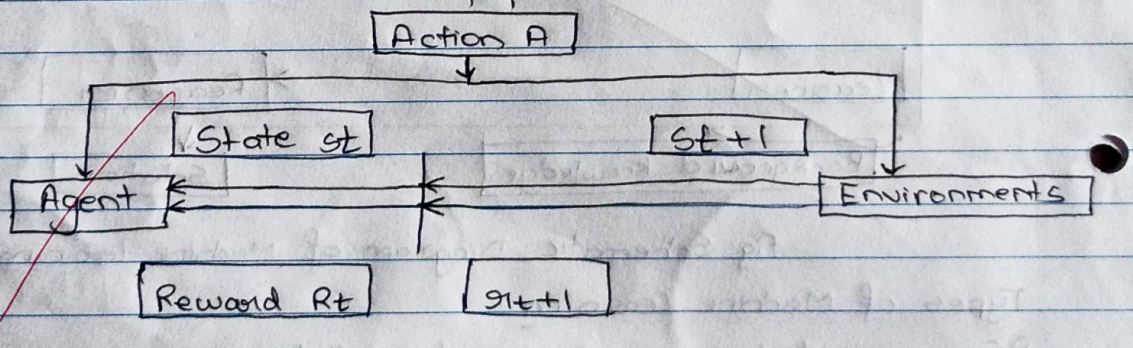
1) Supervised learning - In this type of learning we use data which comprises of input and corresponding output. In supervised learning training data is labelled with correct answers, eg spam or ham. Two most important types of supervised learning are classification and regression.



2) Unsupervised Learning - In unsupervised learning you are only given input x there is no label to the data and given the data or different data points, you may want to form cluster or find some pattern. Two important unsupervised learning tasks are dimension reduction and clustering.



3) Reinforcement Learning - In reinforcement learning you have an agent who is acting in an environment and you want to find out what action the agent must take based on the reward or penalty that the agent gets. In this a agent, eg a robot or controller seeks to learn the optimal actions based on the outcomes of past actions.



4) Semi Supervised Learning - It is a combination of supervised and unsupervised learning. In this there is some amount of labelled training data and also you have a large amount of unlabelled data and you try to come up with some learning algorithm that converts even when training data is not labelled. eg- text classification, image classification, where small amount of labelled data and large amount of unlabelled data is used.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q2 Discuss how to evaluate a ML model for overfitting or underfitting, explain using diagram? What measures need to be taken in case of overfitting and underfitting?

Ans 2 Overfitting - It occurs when a learning model learns the training data too well, capturing noise and details that do not generalise to new data. This leads to excellent performance on training data but poor performance on unseen data.

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

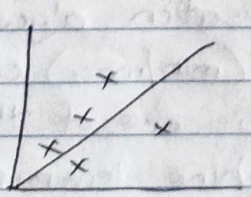
Evaluating a ML model for overfitting and underfitting using training error v/s validation error.

Plotting the training error and validation error against the complexity of the Model (eg- number of parameters or depth of a tree) can help diagnose overfitting and underfitting.

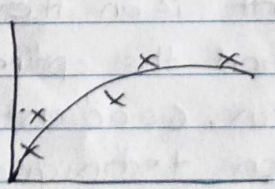
Inunderfitting both training and validation errors are high.

In overfitting training error is low but validation error is high.

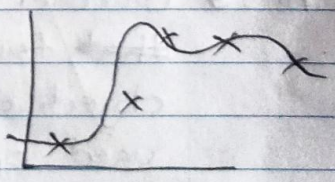
Good fitting - Both training and validation errors are low and close to each other.



Underfitting
(high bias)



Good fitting
(Low bias, Low variance)



Overfitting
(High Variance)

Measures to address overfitting and underfitting-
Overfitting - Simplify the model by reducing the model complexity (eg - fewer parameters, shallowed trees), Regularisation by adding penalties for larger coefficients (eg L1 or L2 regularisation), cross validation techniques to ensure model generalizes well to unseen data, for decision trees, prune the tree to prevent from becoming too deep and stop training the model when performance on the validation set starts to deteriorate.
Underfitting - Increase model complexity by using a complex model that can capture the underlying patterns, create more relevant features that can help the model capture the underlying pattern, reduce regularisation as if regularisation is too strong it can prevent the model from fitting training data well and increase the training time. Sometimes training the model for a longer period can help it learn better.

Q 3 Illustrate process of learning with the gradient descent for a simple linear regression using a bell shaped error curve. Function. Explain how a step size is modulated on every iteration.

Ans 3 Gradient descent is an iterative optimisation algorithm that ~~that~~ tries to find the optimum value (min/max) of an objective function. Gradient descent is one of the most used optimisation techniques in machine learning projects for updating the parameters of a model in order to minimize a cost function. In univariate linear regression, the model has relationship between a single feature x

and a target variable of using a linear equation.

$$y = \theta_0 + \theta_1 x$$

where θ_0 is intercept, θ_1 is slope of the line. For linear regression, the cost function $J(\theta)$ is typically the mean squared error (MSE)

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h\theta(x_i) - y_i)^2$$

Where M is number of training examples

$h(n_0) = \theta_0 + \theta_1 x_i$ is a hypothesis function.

The gradient descent algorithm updates the parameters θ_0 and θ_1 iteratively to minimize $J(\theta)$.

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_j}$$

where α is the learning rate,

$\frac{\partial J(\theta)}{\partial \theta_j}$ is the partial derivative of the cost function with respect to θ_j.

Steps required in Gradient Descent Algorithm are-

Step 1 - Initialize the parameters of the model randomly.

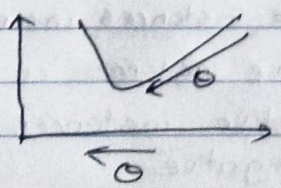
Step 2 - Compute the gradient of the cost function with respect to each parameter.

Step 3 - Update the parameters of the model by taking steps in opposite direction of the model.

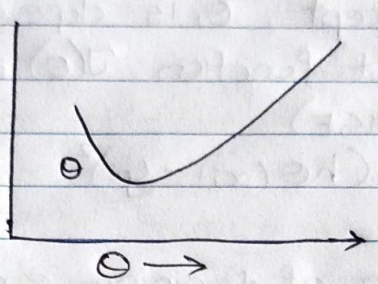
Step 4 - Repeat steps 2 and 3 iteratively to get best parameter for the defined model

Bell shaped error curve for gradient descent -

- slope +ve : $\theta_j = \theta_j$ (+ve value), value of θ_j decreases



Slope -ve:- $\theta_j = \theta_j - (-ve\ value)$. Hence the value of θ_j increases.



Q4 Explain following performance evaluation parameters with the help of confusion matrix.
Illustrate using appropriate example.

1) Accuracy
2) Precision
3) Recall
4) F1 - Score
5) Specificity
6) ROC - AUC curve

Ans 4 A confusion matrix is a table used to describe the performance of a classification model. It compares the predicted outcomes with the actual outcomes. The confusion matrix for a binary classification problem is usually structured like this

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

True Positives (TP) - The number of positive instances correctly predicted.

False Positives (FP) - Number of negative instances incorrectly predicted as positive.

True Negatives (TN) - Number of negative instances correctly predicted.

False Negatives (FN) - Number of positive instances incorrectly predicted as negative.

1) Accuracy - Accuracy is the ratio of correctly predicted observations to the total observations. It gives an overall effectiveness of the model.

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

2) Precision - Also known as Positive Predictive Value, is the ratio of correctly predicted positive observations to the total predicted positives.

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

3) Recall - Also known as Sensitivity or True Positive Rate (TPR) is the ratio of correctly predicated positive observations to all observations in the actual class.

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

4) F1 Score - It is the harmonic mean of precision and recall. It is usefull when you need to balance the two or when you have an uneven class distribution.

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

5) Specificity - Also known as True Negative Rate (TNR) is the ratio of correctly predicted negative observations to all observations in the actual negative class.

$$\text{Specificity} = \frac{TN}{TN+FP}$$

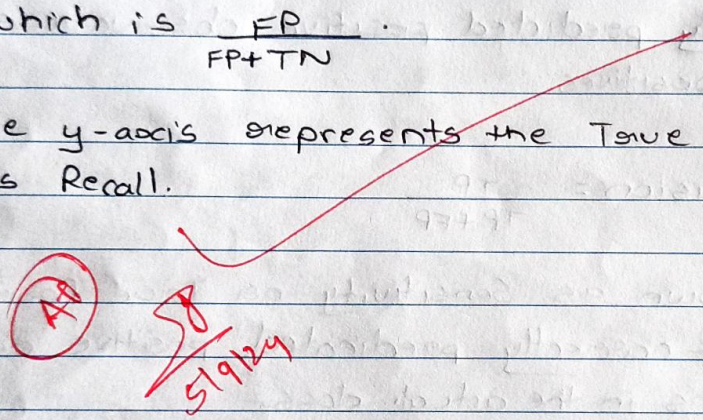
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

6) ROC - AUC Curve -

The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The AUC is the measure of the ability of ~~the~~ a classifier to distinguish between classes.

The x-axis represents the False Positive Rate (FPR) which is $\frac{FP}{FP+TN}$.

The y-axis represents the True Positive Rate which is Recall.



A+8

58/5/9/24

Arnav Malvia
2103109
C32
ML
Group : 44

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment :- 2

Q1 Discuss different ensemble learning techniques.

Ans 1 Ensemble learning is a powerful machine learning technique that combines multiple models to create a more robust and accurate predictive model. The idea is that by combining the strengths of multiple models, the ensemble can perform better than any single model on its own.

1) Bagging - Bagging creates multiple models by training them on different subsets of data, which are generated by random sampling with replacement. This technique, as used in Random Forests, reduces variance and helps prevent overfitting, making models more robust.

2) Boosting - Boosting builds models sequentially, where each new model tries to correct the errors of its predecessor. This technique, seen in Algorithms like AdaBoost and xGBoost, focuses on difficult to predict instances, potentially leading to highly accurate models but also increasing the risk of overfitting.

3) Stacking - Stacking combines predictions from different types of models by training a meta-learner on these predictions. This method leverages the strengths of diverse models, potentially improving overall performance, but it requires careful tuning to avoid complexity and overfitting.

4) Voting - Voting aggregates predictions from multiple models by either majority vote or averaging probabilities. This simple method can enhance model performance by balancing out individual errors, especially when the base models are diverse.

5) Blending - Blending is a simpler alternative to stacking, where base models are trained on one set of data, and their predictions are used to train a meta learner on a seperate validation set. While easier to implement, blending might not fully utilize all the data, which can limit its effectiveness.

6) Bagged Boosting - Bagged boosting combines the principle of bagging and boosting by training multiple boosted models on different data subsets. This method aims to reduce overfitting while maintaining the accuracy benefits of boosting, though it is computationally intensive.

Q2 Explain following terms:-
1) Weak learners and Strong Learners
2) Meta Learning
3) Random Forest.

Ans 2 1) Weak and Strong Learners:-
Weak Learners - These are models that perform slightly better than random guessing but are not very accurate on their own. An example is a simple decision stump. They are often used in ensemble methods like boosting, where multiple weak learners are combined to create a stronger model.

Strong Learners - These are models that perform well on their own and have high accuracy. Strong learnes can solve complex problems without relying on ensemble methods to improve their performance, but they may not still benefit from ensemble techniques to enhance robustness or reduce overfitting.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2) Meta Learning - Meta learning often referred to as "learning to learn" is a process where a model learns from the outputs or patterns of other models, rather than directly from raw data. In ensemble learning, this typically involves training a meta learner to combine the predictions of several base models, as seen in techniques like stacking.

3) Random Forest - Random forest is an ensemble technique designed to combine several decision trees to reduce errors and to build a more accurate prediction model. In random forest instead of building one decision tree, multiple decision trees are built. Each tree in the forest is built using random datapoints drawn from the dataset. The trees in the forest are split such that a fixed subset of input variables are taken each time and the best possible split is performed. Finally, the result from each tree is aggregated to give the outcome.

Q3 Compare Bagging, Boosting and stacking techniques.

Ans 3

	Bagging	Boosting	Stacking
Purpose	Reduce Variance	Reduce Bias	Improve Accuracy
Base Learner Type	Homogeneous	Homogeneous	Heterogeneous
Base Learner Training	Parallel	Sequential	Meta Model
Aggregation	Max Voting, Averaging	Weighted Averaging	Weighted Averaging

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q4 Explain AdaBoost algorithm.

Ans 4 AdaBoost is a boosting algorithm that also works on the principle of the stagewise addition method where multiple weak learners are used for getting strong learners. The value of alpha parameter, in this case will be indirectly proportional to the error of the weak ~~learner~~ learner, Unlike Gradient Boosting in XGBoost the alphaparameter calculated is related to the errors of the weak learner, here the value of the alpha parameter will be indirectly proportional to the error of the weak learner.

Steps involved in AdaBoost:-
1) Weight Initialization
2) Model Training
3) Weighted Error Calculation
4) Model Weight Calculation
5) Update Instance Weights
6) Repeat
7) Final Model Creation
8) Classification.

Advantages of AdaBoost:-
1) Improved Accuracy
2) Versatality
3) Feature Selection
4) Resistance to Overfitting.

Limitations of AdaBoost:-
1) Sensitivity to Noisy Data
2) Computationally Intensive
3) Overfitting
4) Model Selection

Arnav Malula
C32 - 2103109
ML
Group 44

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Assignment :- 3

Q1 Compare decision tree classification with K-NN classification.

Ans1

Decision Tree	KNN
1) Require a proper training phase.	1) Does not require specific training.
2) Builds a model based on training.	2) Zero learning, that is why called lazy algorithm.
3) Once training is completed training data is not needed.	3) Data must always be available for making predictions.
4) It is also interpretable since we can see decision making in tree format.	4) Most interpretable algorithm.
5) Once training is completed, prediction time is less.	5) Takes much time in decision making since it has to traverse the whole dataset to calculate distance with each datapoint.
6) Require initial training time to create decision nodes and branches.	6) Since no ~~time~~ training is required hence the training time is zero.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q2 Compare decision tree classification with logistic regression classification.

Ans 2

Decision Tree	Logistic Regression
1) More interpretable	1) Less interpretable
2) Prone to overfitting.	2) Not prone to overfitting.
3) Majorly affected by noise.	3) Robust to noise.
4) Can be trained on small training set.	4) Requires a large enough training set.
5) Automatically handles decision making.	5) A decision threshold has to be set.

Q3 List down the attribute selection measures used by ID3 algorithm to construct a Decision Tree.

Ans 3 The most widely used algorithm for building a Decision tree is called ID3. ID3 used Entropy and Information Gain as attribute selection measures to construct a Decision Tree.

Entropy - A decision tree is built top-down from a root node and involves the partitioning of data into homogeneous subsets. To check the homogeneity of a sample, ID3 uses entropy. Therefore, entropy is zero when the

sample is completely homogeneous and entropy of one when the sample is equally divided between different classes.

Information Gain - Information gain is based on the decrease in entropy after splitting a dataset based on an attribute. The meaning of constructing a Decision Tree is all about finding the attributes having the highest information gain.

Q4 Explain the properties of Gini Index.

Ans 4 Gini Index is a proportion of impurity or inequality in statistical and monetary settings. In machine learning it is utilized ~~in~~ as an impurity measure in decision tree algorithms for classification tasks.

Properties :-

1) Impurity Measure - The Gini index quatifies how mixed the classes are in a node. A Gini index of 0 indicates that all elements belong to a single class (pure node), while a higher Gini ~~index~~ index indicates more class diversity (impure node).

2) Range - The Gini index ranges from 0 to 0.5 for a binary classification problem. A value of 0 indicates perfect purity while 0.5 indicates maximum impurity. For multiclass problems the maximum value of Gini index can approach 1.

3) Symmetry - Gini index is symmetric with respect to class labels. This means that swapping the class labels does not affect the Gini index which is why it is a popular choice for binary and multi-class classification.

4) Preference for larger partitions - When splitting nodes the Gini index tends to favour larger partitions if they lead to purer nodes, which can make it more sensitive to the distribution of data. This preference helps in growing decision trees that generalize well.

5) Non-Differentiable - Unlike some other impurity measures, Gini index is non-differentiable which makes it a less obvious choice for optimisation in gradient based methods.

6) Computational Efficiency - Gini index is computationally simpler and faster to calculate compared to other impurity measures like entropy, making it a common choice in decision tree algorithms particularly in large datasets.

Q5 Discuss in brief pruning in Decision trees.

Ans 5 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 generalise to new data.

Two main types of decision tree pruning:-

1) Pre-Pruning -

Sometimes the growth of decision tree 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 poor performance when exposed to new data.

Pre-Pruning techniques:

Maximum Depth - Limits the maximum level of depth in a decision tree.

Minimum samples per leaf - set a minimum threshold for the number of samples in each leaf node.

Minimum samples per split - Specify the minimal number of samples needed to break up a node.

Maximum features - Restrict the quantity of features considered for splitting.

2) Post - Pruning -

After the tree is fully grown post pruning involves removing branches or nodes to improve the model's ability to generalize.

Post-Pruning techniques:

Cost Complexity Pruning (CCP) - This method assigns a price to each subtree primarily based on its accuracy and complexity, then selects the subtree with the lowest fee.

Reduced Error Pruning - Removes branches that do not significantly affect the overall accuracy.

Minimum Impurity Decrease - Prunes nodes if the decrease in impurity is beneath a certain threshold.

Minimum Leaf Size - Removes leaf nodes with fewer samples than a specified threshold.

A 8/9/24

Amar Malviya
C32 – 2103109
ML
Group 44

ML

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment :- 4

Q1 Explain the intuition behind logistic regression in detail. Is the decision boundary Linear or Non Linear in the case of a Logistic regression model? Also explain the impact of outliers on Logistic regression?

Ans 1 Logistic Regression is a classification algorithm used when the dependent variable is categorical. The goal of logistic regression is to find the probability that a given input points belongs to a particular class. Unlike ~~the regression~~ linear regression, which predicts continous values, logistic regression predicts probabilities, which are then mapped to binary outcomes (0 or 1).

Steps:-

1) Linear combination of features -

Logistic regression starts similarly to linear regression by calculating a linear ~~calculation~~ combination of input features.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Here β_0 is the intercept, $\beta_1, \beta_2 \ldots, \beta_n$ are the coefficients and x_1, x_2, x_n are the input features.

2) Sigmoid Function -

The key difference is that the output of this linear combination, z, is passed through a sigmoid function to map it to a probability value between 0 and 1.

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

3) Decision Rule -

Based on the probability obtained from the sigmoid function a threshold is applied (commonly 0.5) to classify the input to one of two classes:

if sigmoid $(z) \geq 0.5$, predict class 1; otherwise, predict class 0.

Decision Boundary:

Linear Decision Boundary -

In the case of a simple logistic regression model, the decision boundary is linear. This is because the model is essentially solving for the condition $z = 0$, which represents a hyperplane in the feature space. Therefore, logistic regression assumes that the classes can be seperated by a straight line, a plane or a hyperplane in higher dimensions.

Non-Linear Decision Boundary -

If the input features are transformed, the decision boundary can become non-linear. For example if, you include interaction terms or higher order polynomials in your features, the resulting decision boundary will no longer be a simple straight line.

Impact of outliers on logistic regression:

1) Distorted coefficients - Outliers can skew the model's coefficients, leading to a less accurate decision boundary.
2) Misleading Probabilities - Outliers may cause the model to produce unreliable probability estimates, increasing misclassification risk.
3) Reduced Performance - Overall model performance, including

accuracy, can decline due to outliers affecting the generalizability.

4) Boundary Shift - The decision boundary may be improperly adjusted, favouring outliers over the majority of data.

Q2 What are assumptions made in Logistic Regression? Can we solve the multiclass classification problems using Logistic Regression? If Yes then How?

Ans 2 Assumptions in Logistic Regression:-

1) Linearity of the Logit - The relationship between the independent variables and the log odds of the dependent variable is linear.

2) Independence of observations - The observations should be independent of each other.

3) No or Little Multicollinearity - Independent variables should not be highly correlated with each other.

4) Large Sample Size - Logistic regression requires a large sample size to produce reliable results.

Yes, logistic regression can be extended to handle multiclass classification problem using the following approaches-

1) One-vs-Rest (OvR) - This method involves training multiple binary classifiers, one for each class against all other classes. The class with the highest probability is selected as the prediction.

2) One-vs-One (OvO) - This method involves training binary classifiers for every pair of classes. The class that wins the most pairwise comparisons is chosen as the prediction.

3) Softmax Regression - This is a direct ~~extension~~ extension of logistic regression for multiclass classification, where the model predicts the probability of each class and selects the one with the highest probability.

Q3 Why is Logistic Regression termed as regression and not as Classification?

Ans 3 1) Modeling Approach - Logistic regression models the relationship between the independent variables and the log odds of the dependent variable. The term regression refers to this process of estimating the parameters in a linear equation.

2) Linear combination of features - It uses a linear combination of the input features to predict a continuous value, similar to how linear regression predicts a continuous outcome. This continuous value is then transformed into a probability using the sigmoid function.

3) Historical Naming Convention - Historically, logistic regression was developed as a type of regression model, with the primary goal being to model binary outcomes. The name reflects its origins and the fact that it uses a regression based approach to achieve classification.

4) Probability Estimation - Logistic regression first estimates probabilities, which are then used to classify data into categories. This process of predicting a continuous probability and then making a decision aligns more closely with the concept of regression.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q4 Can we use Mean Square Error as a cost function for logistic regression? Justify your answer.

Ans4 No, Mean Square Error should not be used as a cost function for logistic regression. Here's why:

1) Non-linearity of the sigmoid function - Logistic regression uses the sigmoid function to map predicted values to probabilities. The relationship between the linear predictors and the output is non-linear. Using MSE, which is more suitable for linear relationships, can lead to a non-convex cost function in logistic regression, making it difficult to optimize and leading to poor convergence properties.

2) Inappropriate Error Measurement - MSE is designed for measuring the difference between continuous values, but logistic regression is about classification probabilities. The squared differences in MSE do not appropriately capture the difference between predicted probabilities and the actual binary outcomes.

3) Gradient Issues - MSE can lead to gradients that do not appropriately guide the model toward better classifications, especially when the predicated probabilities are far from the true labels (0 or 1). This can result in slow or ineffective learning.

4) Log-Loss (Cross Entropy) is Better - Logistic regression uses log-loss because it is specifically designed for binary classification tasks. It provides a convex cost function, ensuring that gradient based optimisation methods can efficiently find the global minimum.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q5 Compare Naive Baysian with Logistic regression classifier.

Ans 5

Aspect	Naive Bayes	Logistic Regression
1) Model Type	Generative	Discriminative
2) Underlying Assumptions	Assume features are independent of each other.	No assumptions about independence.
3) Model Complexity	Relatively simple, computationally efficient	More complex, allowing for modeling intricate relationships.
4) Handling Categorical Data	Well suited for categorical data.	Versatile, can handle both numerical and categorical.
5) Interpretability	Highly interpretable due to simplicity and assumption of feature independence.	Good interpretability, through coefficients, indicating strength and direction of feature relationships
6) Robustness to irrelevant features	Can be robust to irrelevant features due to assumption of independence.	May be sensitive to irrelevant features. Regularisation techniques can help mitigate this sensitivity.
7) Data Size and Sparsity	Performs well with small datasets.	May require a larger dataset to avoid overfitting.

Aman Malvia
2103109
C32
ML
Group 44

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment :- 5

Q1 Discuss the need of support vector machines? Explain why they are called as optimal binary classifiers?

Ans1 Need for Support Vector Machines (SVMs):

1) Effective in High Dimensional Spaces - SVMs are particularly effective in cases where the number of features is larger than the number of samples, which is common in text classification and bioinformatics. They can handle high dimensional data well and are effective in both linearly and non linearly seperable data.

2) Robust to Overfitting - SVMs are designed to maximise the marging between different classes, which helps in reducing the chances of overfitting, especially when using the right kernel and regularisation techniques.

3) Versatility with Kernels - SVMs can handle non-linearly seperable data by using kernel tricks (eg- polynomial, radial basis function). These kernels implicitly map input data into higher dimensional spaces, making it easier to find a hyperplane that can seperate the classes.

4) Outlier Resistance - SVMs are relatively robust to outliers because they only rely on a subset of training data, called support vectors, to define the decision boundary. This makes SVMs less sensitive to outliers compared to other models like logistic regression.

SVMs are called optimal Binary classifiers:

1) Maximum Margin Principle - SVMs are called optimal binary classifiers because they are designed to find the hyperplane that maximizes the margin between the two classes. The

margin is the distance between the hyperplane ~~that~~ and the closest data points from each class. Maximising this margin ensures that the classifier has the greatest possible buffer against ~~eee~~ misclassification, making it optimal in terms of classification performance.

2) Support Vectors - The decision boundary in SVM is determined by the support vectors, which are the data points ~~ed~~ closest to the hyperplane. These support vectors are crucial because they define the optimal seperating hyperplane, leading to a more robust and generalizable model.

3) Convex Optimization - The optimization problem in SVMs is convex, meaning there is a single global minimum. This guarantees that the solution found by the SVM is optimal, minimizing the risk of finding a suboptimal solution that could lead to poor classification performance.

4) Generalisation Capability - By maximising the margin, SVMs aim to improve the model's generalisation capability, ensuring it performs well not just on the training data but also on unseen data. This ability to generalise makes SVM's optimal for binary classification tasks.

~~Q2 What are the assumptions~~

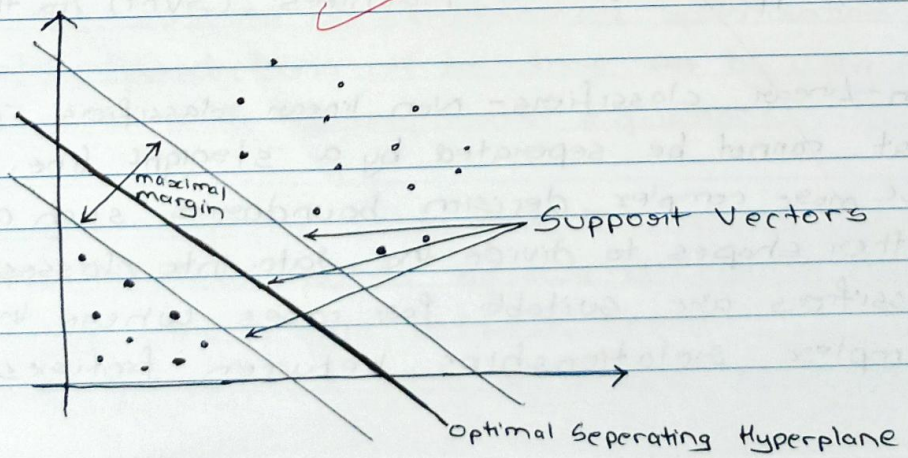
Q2 Explain the following terminologies with the help of appropriate illustrations:
i) Optimal Decision Boundary ii) Support Vectors iii) Margins.

Ans2 i) Optimal Decision Boundary - The optimal decision boundary is the hyperplane that best seperates data points of different

classes in a classification problem. In SVM, the goal is to find the hyperplane that maximizes the margin between the two classes. This boundary is considered optimal because it achieves the best seperation with the maximum margin, which helps the model generalize better to new data.

ii) Support Vectors - Support vectors are the data points that lie closest to the decision boundary (hyperplane) and are the most critical for defining it. These points directly influence the position and orientation of the optimal decision boundary. Even if other data points are removed or changed, as long as the support vectors remain the same, the decision boundary will not change. In essence, support vectors are the key points that "support" the formation of the optimal boundary.

iii) Margins - The margin refers to the distance between the decision boundary (hyperplane) and the closest data points from either class. SVM aims to maximise this margin because a larger margin indicates a more robust classifier that is less sensitive to small changes in the data. A wide margin helps ensure better generalisation, reducing the risk of misclassification on new, unseen data.



Q3 What do you understand by linear classifiers and non linear classifiers? Can you use SVM as non-linear classifier? If yes, explain how a SVM can be used as a non linear classifier.

Ans 3 Linear classifiers - A linear classifier is a model that makes predictions based on a linear combination of input features. The classifier aims to find a straight line that can seperate the data into different ~~cases~~ classes. The decision boundary created by the classifier is linear, meaning the classes can be divided by a straight line. The equation of the decision boundary typically looks like:

$$w_1x_1 + w_2x_2 + \dots + w_nx_n + b = 0$$

This implies that the classifier uses weighted sums of the input features to predict the class label. Linear classifiers work well when the data is linearly seperable, meaning the classes can be seperated by a straight line. However, for complex patterns where data points are not linearly seperable, linear classifiers many not be sufficient. Common examples include Logistic Regression and Support Vector Machines (SVM) in their linear form.

Non-Linear classifiers - Non linear classifiers can handle data that cannot be seperated by a straight line. These classifiers use more complex decision boundaries such as curves or other shapes to divide the data into classes. Non linear classifiers are suitable for cases where there are complex relationships between features and the target

class, and the data is not linearly seperable. Unlike linear classifiers, they can capture these complex patterns by applying transformations to the data or using models with non-linear capabilities. Examples of non linear classifiers include Decision Trees, Neural Networks and Support Vector Machines (SVM) with a Kernel trick. Non-linear classifiers allow for more flexible decision boundaries which adapt to the shape of the data distribution.

SVM as a Non Linear Classifier-

SVM can be extended to work as a non-linear classifier through the use of Kernel functions. When data is not linearly seperable, SVM uses a kernel trick to map the input features into a higher dimensional space where it becomes linearly seperable. This allows SVM to create a linear decision boundary in the higher dimensional space, which translates into a non linear boundary in the original feature space. Examples of kernels used for this purpose include the Radial Basis Function (RBF) and polynomial kernels. This enables SVM to classify complex, non-linear data effectively.

Q4 Express the SVM as a constrained optimisation problem. Discuss how predictions can be done ~~usr~~ by using SVM. Support your answer with appropriate equations.

Ans 4 In SVM the objective is to find the optimal hyperplane that seperates the data points of two different classes while maximizing the margin between them.
This can be formulated as a contrained optimisation problem.

Objective Function:

The goal is to minimize the norm of the weight vector w, which is equivalent to maximising the margin between the two classes. The optimisation is defined as:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

Here w is the weight vector and b is the bias term. Minimizing $\|w\|^2$ ensures the margin is maximised.

Constraints:

To ensure that the data points are correctly classified, we add the following constraints for each data point (x_i, y_i):

$$y_i (w^T x_i + b) \geq 1 \; \forall i$$

where

x_i is the feature vector for the i^{th} data point.

y_i is the label (either +1 or -1) of the i^{th} data point.

$w^T x_i + b$ represents the linear decision boundary.

This constraint ensures that all points are classified correctly and the margin between the classes is atleast 1 unit.

Thus the complete constrained optimisation problem is:

$$\min_{w,b} \frac{1}{2} \|w\|^2 \text{ subject to } y_i (w^T x_i + b) \geq 1, \forall i$$

Predictions in SVM:

Once the optimal values for w and b have been found by solving the above optimisation problem, the SVM model can be used to make predictions. The decision rule for predicting the class of a new data point x is given by:

$$f(x) = \text{sign}(w^T x + b)$$

Explanation:

- $w^T x + b$ is the distance of the point x from the hyperplane
- The sign function outputs:
  - +1 if x lies on the positive side of the hyperplane (predicts class is +1)
  - -1 if x lies on the negative side of the hyperplane (predicts class is -1)

This decision rule classifies the new data point by checking on which side of the hyperplane it lies if $f(x) > 0$ the point belongs to the positive class, otherwise it belongs to the negative class.

Q5 What ~~is~~ are Kernel functions? List some kernel functions. What is their use in SVM?

Ans 5 Kernel functions are mathematical functions that transform data into a higher dimensional space, making it easier to seperate non-linearly seperable data. By applying a kernel function, SVM can create a linear decision boundary in this transformed space, which corresponds to a non linear boundary in the original input space.

The key idea is that rather than working directly in the original feature space, Kernel functions allow SVM to operate in an implicit higher dimensional feature-space without the need to compute the transformation explicitly. This is achieved through the kernel trick, where the dot product between two vectors in the transformed feature space is computed directly using the kernel function without ever computing the actual mapping.

Kernel Functions:

1) Linear Kernel - Used for linearly seperable data, this kernel computes the dot product between the feature vectors without transforming them into a higher dimensional space.
2) Polynomial Kernel - Suitable ~~da~~ for data that can be seperated by polynomial decision boundaries. The degree d controls the flexibility of the decision boundary.
3) Gaussian Kernel (Radial Basis Function) - One of the most popular kernels used for non linearly seperable data. It measures the similarity between two points based on their distance, with γ controlling the spread of the kernel.
4) Sigmoid Kernel - Similar to the activation function in neural networks, this kernel is used in certain scenarios where the relationship between the data points is similar to that modeled by neural networks.

Use of Kernel functions in SVM:

1) Non-Linear Data - When the data cannot be seperated by a straight line, Kernel functions map the input data into a higher dimensional space where it may become linearly seperable.
2) Efficient Computation - Instead of explicitly transforming the data into a high dimensional space, the kernel trick allows SVM to calculate the dot products directly in this space.
3) Flexibility - By using different kernel functions, SVM can adapt to various types of data, enabling it to handle complex decision boundaries.

Q6 Explain the concept of Kernel trick. Discuss with example, how Kernel tricks can speed up the computations.

Ans 6 The Kernel trick is a mathematical technique used in machine learning algorithms, particularly Support Vector Machines (SVMs), to efficiently compute the dot product in a higher dimensional space without explicitly mapping the data to that space. It enables the model to create non-linear decision boundaries while avoiding the computational cost of transforming the data into higher dimensions.
The Kernel trick relies on the inner product of vectors. For SVMs, the decision function is based on the dot products of vectors within the input space. Kernel functions replace these dot products with a non-linear function that computes a dot product in a higher dimensional space. Importantly, the computation of this dot product via

the Kernel function does not require explicit knowledge of the coordinates in the higher space, thus saving computational resources and time.

Example:
Consider two data points x_1 and x_2 in the original input space. If we wanted to map these points to a higher dimensional space $\phi(x_1)$ and $\phi(x_2)$, computing the transformation directly could be expensive. However using a Kernel function $K(x_1, x_2)$, we can calculate the dot product in the higher dimensional space without explicitly performing the transformation:

$$K(x_1, x_2) := \phi(x_1)^T \phi(x_2)$$

Instead of calculating $\phi(x_1)$ and $\phi(x_2)$ seperately the Kernel function directly gives the dot product in the transformed space. This reduces the computational complexity significantly.

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

Ans 7 +ve labelled : (3,1) , (3,-1) , (6,1) , (6,-1)
-ve labelled : (1,0) , (0,1) , (0,-1) , (-1,0)

For linear SVM

$$w_1 x_1 + w_2 x_2 + b = 0$$

For all the points:

$w_1 x_1 + w_2 x_2 + b \geq 1$

For all -ve points:

$w_1 x_1 + w_2 x_2 + b \leq -1$

$y_i(w_1 x_i + w_2 x_{i_2} + b) \geq 1$ for all i

$x_1 + x_2 + b = 0$

using vector $(3,1)$

$3 + 1 + b = 1$

$b = -3$

For -ve class: $(1,0)$

$1 + 0 - 3 = -2$

$x_1 + x_2 - 2 = 0$

classify the point $(1,3)$

$1 + 3 - 2 > 0$

$2 > 0$

Since result is positive (+ve) the point lies on the side.

Q8 Obtain the optimal Binary hyperplane for classifying the data points given below

Positive class :- (1,1), (3,1), (1,4)
Negative class :- (2,4), (3,3), (5,1)

$w_1 x_{1i} + w_2 x_{2i} + b \geq 1$ for +ve
$w_1 x_{1i} + w_2 x_{2i} + b \leq 1$ for -ve

SVM optimisation = $\frac{1}{2} |w|^2$

Consider point (1,1) and (3,3)
Assume $ax_1 + bx_2 + c = 0$ as hyperplane line

For (1,1)
$a(1) + b(1) + c = 1$

For (3,3)
$a(3) + b(3) = -1$

$x_1 - x_2 = 0$
$x_2 = 2x_1 - 2$
$x_1 - 2x_2 + 2 = 0$

The optimal decision boundary that seperates the +ve and -ve classes can be expressed as :-
$x_1 - 2x_2 + 2 = 0$.

7/10/24