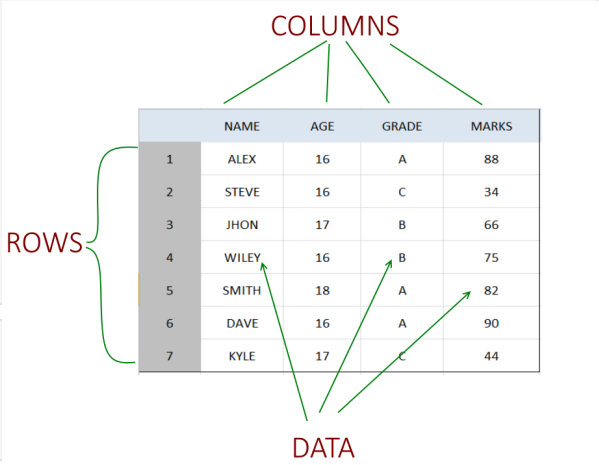
Mastering Step by Step Data Manipulation in Python: A Guide to Effortless Data Analysis

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline


sns.set(rc={"axes.facecolor":"Beige" , "axes.grid" : False})

import warnings
warnings.filterwarnings("ignore")
```



In [2]:
```
# Set Display

pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
pd.set_option('display.precision', 2)
```

In [3]:
```
# Check Library Version

import numpy
print('numpy:{}'.format(numpy.__version__))
```

```
numpy:1.26.4
```

Learn AI from Beginner to Mastery. Click here to register.

In [4]:
```
# Load the dataset

df = pd.read_csv("titanic.csv")
display(df.shape)
df.head()
```

```
(891, 12)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S

In [5]:
```
# Set Table Properties

df.head(3).style.set_properties(**{'background-color': 'blue',
                           'color': 'white',
                           'border-color': 'darkblack'})
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.250000	nan	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.000000	1	0	PC 17599	71.283300	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.925000	nan	S

In [6]:
```
# Replacing Values/Names in a Column:

df1 = df.copy('Deep')
df1["Survived"].replace({0:"Died" , 1:"Saved"},inplace = True)
df1.head(3)
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	Died	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
1	2	Saved	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28	C85	C
2	3	Saved	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S

In [7]:
```
# Drop Columns
df1 = df.copy('Deep')
df1 = df1.drop(['PassengerId','Ticket'],axis=1)
df1.head(3)
```

Out[7]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.25	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.28	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.92	NaN	S

In [8]:
```
# Drop Rows

df = df.drop(labels=[1,3,5,7],axis=0)
df.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S

In [9]:
```
# Missing Value check

print('Method 1:')
df.isnull().sum()
```

Out[9]:
```
Method 1:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            176
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          685
Embarked         2
dtype: int64
```

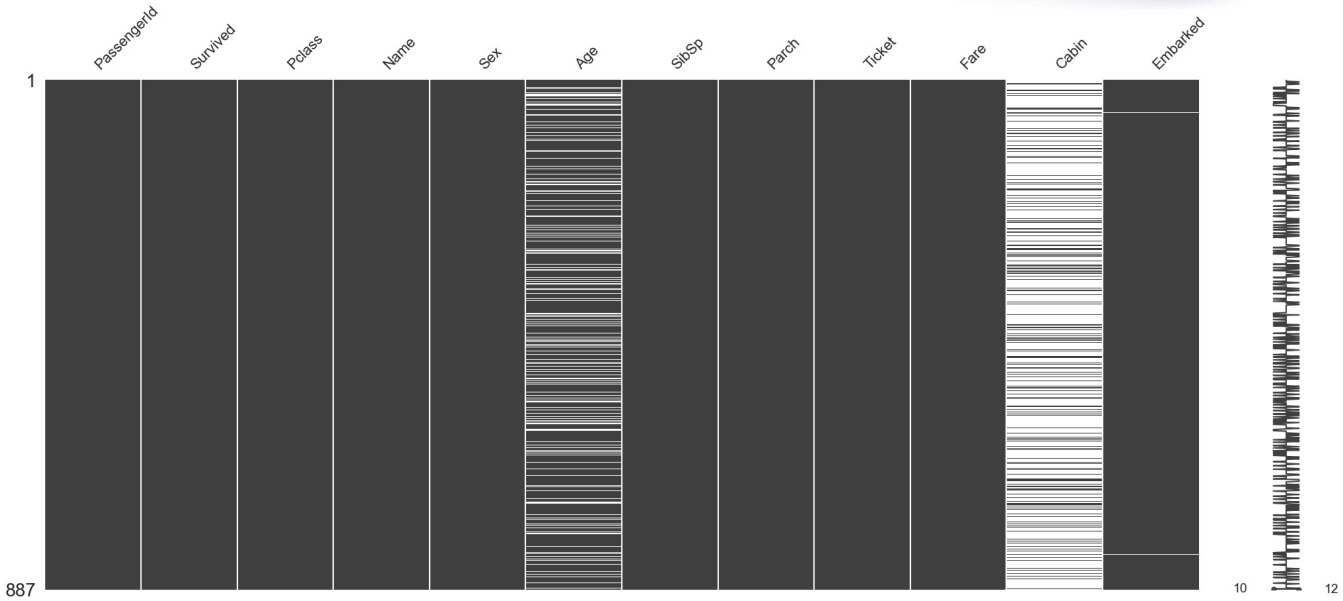
In [10]:
```
var1 = [col for col in df.columns if df[col].isnull().sum() != 0]
print(df[var1].isnull().sum())
```

```
Age         176
Cabin       685
Embarked      2
dtype: int64
```

In [11]:
```
# Missing Value check

print('Method 12:')
import missingno as msno
msno.matrix(df)
plt.show()
```

```
Method 12:
```

Learn AI from Beginner to Mastery. Click here to register.



In [12]:
```
df[df['Embarked'].isnull()]
```

Out[12]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
61	62	1	1	Icard, Miss. Amelie	female	38.0	0	0	113572	80.0	B28	NaN
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	NaN

In [13]:
```
sample_incomplete_rows =df[df.isnull().any(axis=1)].head()
sample_incomplete_rows
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.07	NaN	C

In [14]:
```
df.describe()
```

Out[14]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	887.00	887.00	887.00	711.00	887.00	887.00	887.00
mean	447.99	0.38	2.31	29.72	0.52	0.38	32.18
std	256.22	0.49	0.83	14.52	1.10	0.81	49.78
min	1.00	0.00	1.00	0.42	0.00	0.00	0.00
25%	226.50	0.00	2.00	20.25	0.00	0.00	7.90
50%	448.00	0.00	3.00	28.00	0.00	0.00	14.45
75%	669.50	1.00	3.00	38.00	1.00	0.00	31.00
max	891.00	1.00	3.00	80.00	8.00	6.00	512.33

In [15]:
```
# Describe
df[df['Survived']==0].describe().T.style.background_gradient(subset=['mean','std','50%','count'], cmap='RdPu')
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
PassengerId	547.000000	448.625229	259.750818	1.000000	213.500000	457.000000	675.500000	891.000000
Survived	547.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Pclass	547.000000	2.530165	0.736605	1.000000	2.000000	3.000000	3.000000	3.000000
Age	423.000000	30.693853	14.120135	1.000000	21.000000	28.000000	39.000000	74.000000
SibSp	547.000000	0.550274	1.286281	0.000000	0.000000	0.000000	1.000000	8.000000
Parch	547.000000	0.329068	0.824052	0.000000	0.000000	0.000000	0.000000	6.000000
Fare	547.000000	22.144765	31.440164	0.000000	7.854200	10.500000	26.000000	263.000000

In [16]:
```
df.describe(percentiles=[0.05,0.25,0.35,0.5,0.75,0.85,0.95,0.995,0.999])
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	887.00	887.00	887.00	711.00	887.00	887.00	887.00
mean	447.99	0.38	2.31	29.72	0.52	0.38	32.18
std	256.22	0.49	0.83	14.52	1.10	0.81	49.78
min	1.00	0.00	1.00	0.42	0.00	0.00	0.00
5%	49.30	0.00	1.00	4.00	0.00	0.00	7.22
25%	226.50	0.00	2.00	20.25	0.00	0.00	7.90
35%	315.10	0.00	2.00	24.00	0.00	0.00	9.00
50%	448.00	0.00	3.00	28.00	0.00	0.00	14.45
75%	669.50	1.00	3.00	38.00	1.00	0.00	31.00
85%	758.10	1.00	3.00	45.00	1.00	1.00	56.50
95%	846.70	1.00	3.00	56.00	2.70	2.00	112.56
99.5%	886.57	1.00	3.00	70.73	8.00	5.00	263.00
99.9%	890.11	1.00	3.00	75.74	8.00	5.11	512.33
max	891.00	1.00	3.00	80.00	8.00	6.00	512.33

In [17]:
```
# Agg
df[['Age','Fare','Pclass']].agg(['sum','max','mean','std','skew','kurt'])
```

Out[17]:

	Age	Fare	Pclass
sum	21130.17	28540.03	2049.00
max	80.00	512.33	3.00
mean	29.72	32.18	2.31
std	14.52	49.78	0.83
skew	0.40	4.79	-0.63
kurt	0.18	33.33	-1.27

In [18]:
```
# value_counts
df['Embarked'].value_counts().to_frame()
```

Out[18]:

	count
Embarked	
S	642
C	167
Q	76

In [19]:
```
df['Embarked'].value_counts().tolist()
```

Out[19]:
```
[642, 167, 76]
```

In [20]:
```
# value_counts for Multiple Columns
for col in df[['Survived','Sex','Embarked']]:
    print(df[col].value_counts().to_frame())
    print("****"*7)
```

```
          count
Survived
0           547
1           340
**************************
        count
Sex
male      575
female    312
**************************
          count
Embarked
S           642
C           167
Q            76
**************************
```

In [21]:
```
#Count
df[['Age','Embarked','Sex']].count()
```

Out[21]:
```
Age         711
Embarked    885
Sex         887
dtype: int64
```

In [22]:
```
df['Embarked'][df['Sex']=='female'].value_counts(normalize=True)*100
```

Out[22]:
```
Embarked
S    65.16
C    23.23
Q    11.61
Name: proportion, dtype: float64
```

In [23]:
```
df['Embarked'].value_counts()/len(df['Embarked'])
```

Out[23]:
```
Embarked
S    0.72
C    0.19
Q    0.09
Name: count, dtype: float64
```

In [24]:
```
#Shuffling the data
df2 = df.sample(frac=1,random_state=3)
df2.head()
```

Out[24]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
733	734	0	2	Berriman, Mr. William John	male	23.0	0	0	28425	13.00	NaN	S
95	96	0	3	Shorney, Mr. Charles Joseph	male	NaN	0	0	374910	8.05	NaN	S
161	162	1	2	Watt, Mrs. James (Elizabeth "Bessie" Inglis Mi...	female	40.0	0	0	C.A. 33595	15.75	NaN	S
392	393	0	3	Gustafsson, Mr. Johan Birger	male	28.0	2	0	3101277	7.92	NaN	S
614	615	0	3	Brocklebank, Mr. William Alfred	male	35.0	0	0	364512	8.05	NaN	S

In [25]:
```
df1 = df.copy()

columns = ['Age']

for col in columns:
    df1[col].replace(0, np.NaN, inplace=True)

df1.describe()
```

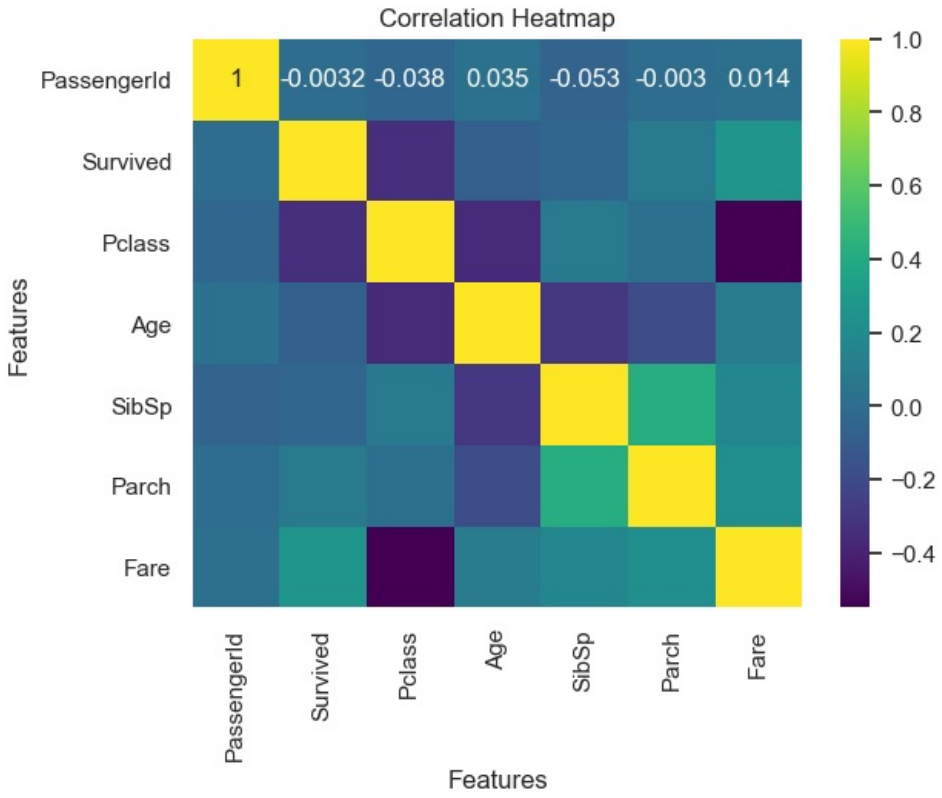
Out[25]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	887.00	887.00	887.00	711.00	887.00	887.00	887.00
mean	447.99	0.38	2.31	29.72	0.52	0.38	32.18
std	256.22	0.49	0.83	14.52	1.10	0.81	49.78
min	1.00	0.00	1.00	0.42	0.00	0.00	0.00
25%	226.50	0.00	2.00	20.25	0.00	0.00	7.90
50%	448.00	0.00	3.00	28.00	0.00	0.00	14.45
75%	669.50	1.00	3.00	38.00	1.00	0.00	31.00
max	891.00	1.00	3.00	80.00	8.00	6.00	512.33

In [26]:
```
corr = df.select_dtypes('number').corr()
display(corr)

sns.heatmap(corr, annot=True, cmap='viridis')

plt.xlabel('Features')
plt.ylabel('Features')
```

```
plt.title('Correlation Heatmap')
plt.show()
```

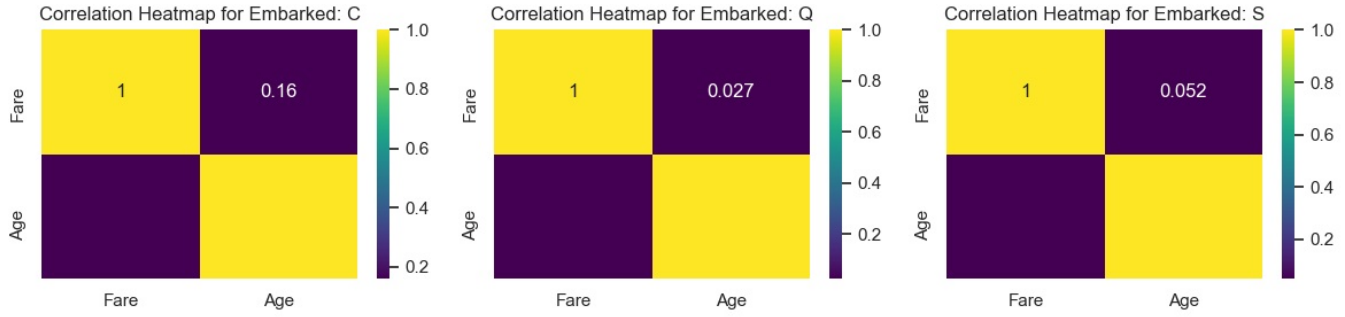
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.00e+00	-3.15e-03	-0.04	0.03	-0.05	-2.95e-03	0.01
Survived	-3.15e-03	1.00e+00	-0.33	-0.08	-0.04	8.35e-02	0.26
Pclass	-3.84e-02	-3.35e-01	1.00	-0.37	0.08	1.66e-02	-0.55
Age	3.49e-02	-8.15e-02	-0.37	1.00	-0.30	-1.87e-01	0.09
SibSp	-5.30e-02	-3.52e-02	0.08	-0.30	1.00	4.15e-01	0.16
Parch	-2.95e-03	8.35e-02	0.02	-0.19	0.41	1.00e+00	0.22
Fare	1.38e-02	2.56e-01	-0.55	0.09	0.16	2.17e-01	1.00



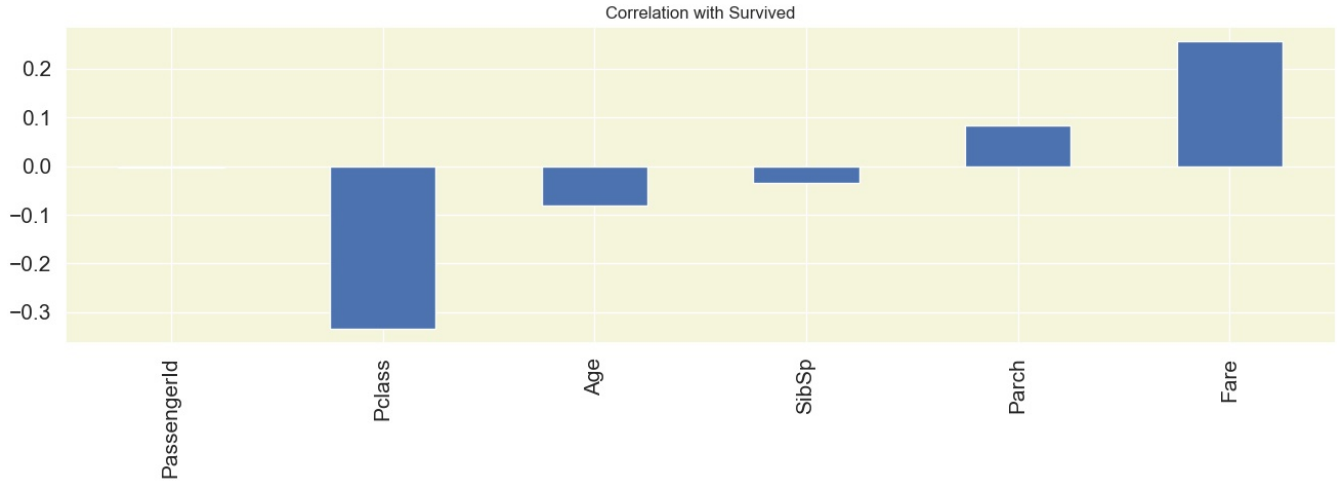
In [27]:

```
corr = df.groupby(["Embarked"])[["Fare", "Age"]].corr()
display(corr)

n_groups = len(corr.index.levels[0])

fig, axes = plt.subplots(nrows=1, ncols=min(3, n_groups), figsize=(12, 3))

group_count = 0
for embarked_group in corr.index.levels[0]:
    ax = axes.flat[group_count]
    sns.heatmap(corr.xs(embarked_group), annot=True, cmap='viridis', ax=ax)
    ax.set_title(f"Correlation Heatmap for Embarked: {embarked_group}")
    group_count += 1

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

		Fare	Age
Embarked			
C	**Fare**	1.00	0.16
	Age	0.16	1.00
Q	**Fare**	1.00	0.03
	Age	0.03	1.00
S	**Fare**	1.00	0.05
	Age	0.05	1.00

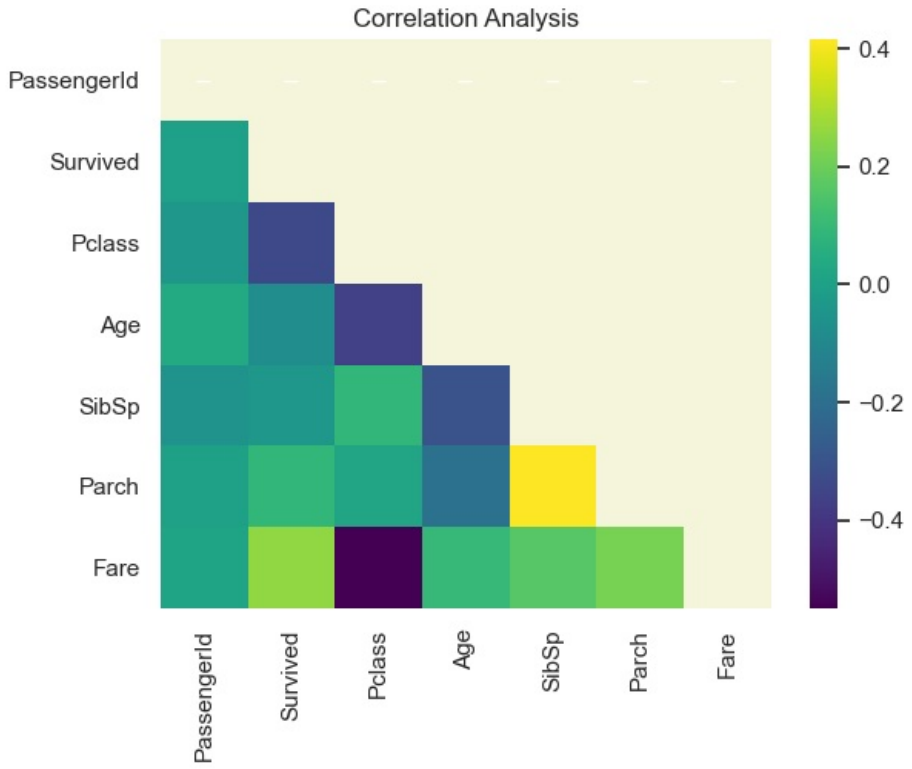


In [28]:
```
X = df.select_dtypes('number').drop(['Survived'],axis=1)
y = df.select_dtypes('number')['Survived']
X.corrwith(y).plot.bar(
        figsize = (16, 4), title = "Correlation with Survived", fontsize = 15,
        rot = 90, grid = True)
plt.show()
```



In [29]:
```
corr = df.select_dtypes('number').corr()
mask = np.triu(np.ones_like(corr,dtype = bool))
plt.figure(dpi=100)
plt.title('Correlation Analysis')
sns.heatmap(corr,mask=mask,annot=True,lw=0,linecolor='white',cmap='viridis',fmt = "0.2f")
plt.xticks(rotation=90)
plt.yticks(rotation = 0)
plt.show()
```

Learn AI from Beginner to Mastery. Click here to reaister.



In [30]:
```
abs(df.select_dtypes('number').corr()).style.highlight_min(axis=0)
```

Out[30]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	0.003153	0.038432	0.034866	0.052966	0.002953	0.013752
Survived	0.003153	1.000000	0.334575	0.081455	0.035186	0.083506	0.255761
Pclass	0.038432	0.334575	1.000000	0.367249	0.083521	0.016574	0.548991
Age	0.034866	0.081455	0.367249	1.000000	0.304297	0.187338	0.094965
SibSp	0.052966	0.035186	0.083521	0.304297	1.000000	0.414724	0.159978
Parch	0.002953	0.083506	0.016574	0.187338	0.414724	1.000000	0.217091
Fare	0.013752	0.255761	0.548991	0.094965	0.159978	0.217091	1.000000

In [31]:
```
df1 = df1[df1['Cabin'].notna()]
df1.head()
```

Out[31]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S

In [32]:
```
print('Passenger Survived in Titanic: {:d}'.format(df1['Survived'].value_counts()[0]))
print('Passenger Died in Titanic: {:d}'.format(df1['Survived'].value_counts()[1]))
```

```
Passenger Survived in Titanic: 68
Passenger Died in Titanic: 134
```

In [33]:
```
print('Survived sample ratio: {:.3f} %'.format(df1['Survived'].value_counts()[0]/len(df1)*100))
print('Died sample ratio: {:.3f} %'.format(df1['Survived'].value_counts()[1]/len(df1)*100))
```

```
Survived sample ratio: 33.663 %
Died sample ratio: 66.337 %
```

In [34]:
```
# Fillna Method

df1 = df.copy()
df1 = df1.dropna()
```

In [35]:
```
# Fillna Method

df1 = df.copy()
df1.fillna(method="ffill", inplace=True)
```

In [36]:
```
# Fill Null Values by Mean Value

df1 = df.copy()
```

```
df1["Age"] = df1["Age"].fillna(df1["Age"].mean())
```

In [37]:
```
# Fill Null Values by Desiresd Value

df1 = df.copy()
df1['Embarked'] = df1['Embarked'].fillna(df1['Embarked'] == 'Q')
df1.head(3)
```

Out[37]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S

In [38]:
```
#Fill Method :
df1 = df.copy()
df1['Age'] = df1['Age'].fillna(0)
df1['Age'] = df1['Age'].fillna('None')
df1["Age"].fillna(method="backfill",inplace=True)
df1["Embarked"].fillna(value="A",inplace=True)
df1["Pclass"].fillna(value= 0,inplace=True)
df1.head()
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S

In [39]:
```
# Find Method/Select Method

# Find All Null Values in the dataframe

df1 = df.copy()
df1 = df1.drop('Cabin',axis =1)

sample_incomplete_rows = df1[df1.isnull().any(axis=1)]
display(sample_incomplete_rows.shape)
sample_incomplete_rows.head()
```

```
(178, 11)
```

Out[39]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.00	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.22	C
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.22	C
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.88	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.90	S

In [40]:
```
titanic_Fare500 = df1[df1['Fare'] > 500][['Name','Embarked']]
display(titanic_Fare500.shape)
titanic_Fare500
```

```
(3, 2)
```

Out[40]:

	Name	Embarked
258	Ward, Miss. Anna	C
679	Cardeza, Mr. Thomas Drake Martinez	C
737	Lesurer, Mr. Gustave J	C

In [41]:
```
titanic_age_70 = df1.loc[df1['Age'] > 70, ["Name","Embarked","Sex"]]
display(titanic_age_70.shape)
titanic_age_70
```

```
(5, 3)
```

Out[41]:

	Name	Embarked	Sex
96	Goldschmidt, Mr. George B	C	male
116	Connors, Mr. Patrick	Q	male
493	Artagaveytia, Mr. Ramon	C	male
630	Barkworth, Mr. Algernon Henry Wilson	S	male
851	Svensson, Mr. Johan	S	male

In [42]:
```
titanic_age_selection = df[(df["Sex"] == "male") & (df["Age"] > 50.00)]
display(titanic_age_selection.shape)
titanic_age_selection.head()
```

```
(47, 12)
```

Out[42]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.50	NaN	S
54	55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.98	B30	C
94	95	0	3	Coxon, Mr. Daniel	male	59.0	0	0	364500	7.25	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.65	A5	C

In [43]:
```
women = df1.loc[df1['Sex'] == 'female']["Survived"]
rate_women = (women.sum()/len(women)).round(3)*100
print("Percentage of women who survived:",rate_women,"%")
```

```
Percentage of women who survived: 74.0 %
```

In [44]:
```
men = df1.loc[df1['Sex'] == 'male']["Survived"]
rate_men = (men.sum()/len(men)).round(3)*100
print("Percentage of Men who survived:", rate_men,"%")
```

```
Percentage of Men who survived: 19.0 %
```

In [45]:
```
titanic_Pclass = df1[df1["Pclass"].isin([1, 2])]
display(titanic_Pclass.shape)
titanic_Pclass.head()
```

```
(398, 11)
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.07	C
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.00	S
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.00	S

In [46]:
```
df1 = df.copy()
cabin_no_na = df1[df1["Cabin"].notna()]
display(cabin_no_na.shape)
cabin_no_na.head()
```

```
(202, 12)
```

Out[46]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S

In [47]:
```
titanic_Pclass = df1[(df1["Pclass"] == 1) & (df1["Sex"] == 'female') & (df1["Age"] > 50 ) ]

display(titanic_Pclass.shape)
titanic_Pclass.head()
```

```
(13, 12)
```

Out[47]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
195	196	1	1	Lurette, Miss. Elise	female	58.0	0	0	PC 17569	146.52	B80	C
268	269	1	1	Graham, Mrs. William Thompson (Edith Junkins)	female	58.0	0	1	PC 17582	153.46	C125	S
275	276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.96	D7	S
366	367	1	1	Warren, Mrs. Frank Manley (Anna Sophia Atkinson)	female	60.0	1	0	110813	75.25	D37	C

In [48]:
```
# np.where

df1 = df.copy()
df1['Cabin_null'] = np.where(df1['Cabin'].isnull(),0,1)
df1.head()
```

Out[48]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Cabin_null
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0

In [49]:
```
df1 = df.copy()
df1["Bucket"] = np.where(df1["Fare"] < 250, "Low", "High")
df1.head()
```

Out[49]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Bucket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	Low
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	Low
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	Low
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	Low
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	Low

In [50]:
```
df1 = df.copy()

titanic_age_missing_first  = df1.sort_values(by='Age',ascending=False, na_position='first')
titanic_age_missing_first.head()
```

Out[50]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.00	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.22	NaN	C
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.22	NaN	C
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.88	NaN	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.90	NaN	S

In [51]:
```
df1 = df.copy()
df1.sort_values(by = 'Age' , ascending = False)[['Name','Ticket','Survived','Pclass', 'Age' ]].head()
```

Out[51]:

	Name	Ticket	Survived	Pclass	Age
630	Barkworth, Mr. Algernon Henry Wilson	27042	1	1	80.0
851	Svensson, Mr. Johan	347060	0	3	74.0
96	Goldschmidt, Mr. George B	PC 17754	0	1	71.0
493	Artagaveytia, Mr. Ramon	PC 17609	0	1	71.0
116	Connors, Mr. Patrick	370369	0	3	70.5

In [52]:
```
Numerical_data = df1.select_dtypes(include=['number'])
Numerical_data.head()
```

Out[52]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1	0	7.25
2	3	1	3	26.0	0	0	7.92
4	5	0	3	35.0	0	0	8.05
6	7	0	1	54.0	0	0	51.86
8	9	1	3	27.0	0	2	11.13

In [53]:
```
Categorical_data = df1.select_dtypes(include=['object'])
Categorical_data.head()
```

Out[53]:

	Name	Sex	Ticket	Cabin	Embarked
0	Braund, Mr. Owen Harris	male	A/5 21171	NaN	S
2	Heikkinen, Miss. Laina	female	STON/O2. 3101282	NaN	S
4	Allen, Mr. William Henry	male	373450	NaN	S
6	McCarthy, Mr. Timothy J	male	17463	E46	S
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	347742	NaN	S

In [54]:
```
cabin_notna = df1[df1['Cabin'].notna()]
cabin_notna.head()
```

Out[54]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S

In [55]:
```
#groupby

df1 = df.copy()
titanic_room = df1.groupby(['Embarked'])['Age'].mean().reset_index()
titanic_room.head()
```

Out[55]:

	Embarked	Age
0	C	30.76
1	Q	28.09
2	S	29.49

In [56]:
```
df1.groupby("Embarked").agg({"Fare": np.mean, "Sex": np.size})
```

Out[56]:

	Fare	Sex
Embarked		
C	59.89	167
Q	13.34	76
S	27.05	642

In [57]:
```
temp = df1.groupby("Sex")['Age'].min().to_frame().reset_index()
temp
```

Out[57]:

	Sex	Age
0	female	0.75
1	male	0.42

In [58]:
```
df1.groupby(["Embarked", "Pclass"]).agg({"Fare": [np.size, np.mean]})
```

Out[58]:

		Fare	
		size	mean
Embarked	Pclass		
C	1	84	105.12
	2	17	25.36
	3	66	11.21
Q	1	2	90.00
	2	3	12.35
	3	71	11.22
S	1	126	70.50
	2	164	20.33
	3	352	14.63

In [59]:
```
df1.groupby(['Survived',"Sex"])['Fare'].first().to_frame()
```

Out[59]:

		Fare
Survived	Sex	
0	female	7.85
	male	7.25
1	female	7.92
	male	13.00

In [60]:
```
Tit_groupby = df1.groupby("Pclass")["Pclass"].count().to_frame()
Tit_groupby
```

Out[60]:

	Pclass
Pclass	
1	214
2	184
3	489

In [61]:
```
df1.groupby('Survived')['Sex'].value_counts().to_frame()
```

Out[61]:

		count
Survived	Sex	
0	male	466
	female	81
1	female	231
	male	109

In [62]:
```
df1.groupby(['Survived',"Sex"])['Pclass'].count()/df1.groupby(["Sex"])['Pclass'].count()*100
```

Out[62]:
```
Survived  Sex
0         female    25.96
          male      81.04
1         female    74.04
          male      18.96
Name: Pclass, dtype: float64
```

In [63]:
```
(df1.groupby(['Embarked','Pclass']).count()['Fare']/df1.groupby(['Embarked']).count()['Fare'])*100
```

Out[63]:
```
Embarked  Pclass
C         1         50.30
          2         10.18
          3         39.52
Q         1          2.63
          2          3.95
          3         93.42
S         1         19.63
          2         25.55
          3         54.83
Name: Fare, dtype: float64
```

In [64]:
```
df1.groupby("Sex")[["Age","Pclass"]].mean()
```

Out[64]:

Sex	Age	Pclass
female	27.85	2.17
male	30.79	2.39

In [65]:
```
df1.groupby(["Sex", "Pclass"])["Fare"].mean()
```

Out[65]:
```
Sex     Pclass
female  1         107.08
        2          21.97
        3          16.12
male    1          67.23
        2          19.74
        3          12.65
Name: Fare, dtype: float64
```

In [66]:
```
titanic_summed = df1.groupby(["Sex", "Embarked"])[["Fare", "Age"]].sum()
titanic_summed
```

Out[66]:

Sex	Embarked	Fare	Age
female	C	5416.11	1691.00
	Q	454.86	291.50
	S	7811.31	5130.50
male	C	4584.90	2276.92
	Q	558.94	495.00
	S	9553.92	11145.25

In [67]:
```
df1.groupby(["Embarked", "Pclass"]).agg({"Fare": [np.size, np.mean]})
```

Out[67]:

		Fare	
		size	mean
Embarked	Pclass		
C	1	84	105.12
	2	17	25.36
	3	66	11.21
Q	1	2	90.00
	2	3	12.35
	3	71	11.22
S	1	126	70.50
	2	164	20.33
	3	352	14.63

In [68]:
```
temp = df1.groupby("Sex")['Age'].min().reset_index()
temp
```

Out[68]:

	Sex	Age
0	female	0.75
1	male	0.42

In [69]:
```
titanic_room= df1.groupby(['Embarked','Sex'])[['Age','Fare']].mean().reset_index()
titanic_room
```

Out[69]:

	Embarked	Sex	Age	Fare
0	C	female	28.18	75.22
1	C	male	33.00	48.26
2	Q	female	24.29	12.63
3	Q	male	30.94	13.97
4	S	female	27.73	38.67
5	S	male	30.37	21.71

In [70]:
```
df1.groupby(['Survived',"Sex","Embarked"])['Pclass'].count().to_frame()
```

Out[70]:

Survived	Sex	Embarked	Pclass
0	female	C	9
		Q	9
		S	63
	male	C	66
		Q	37
		S	363
1	female	C	63
		Q	27
		S	139
	male	C	29
		Q	3
		S	77

In [71]:
```
df1.groupby("Embarked").agg({"Fare": np.mean, "Sex": np.size})
```

Out[71]:

Embarked	Fare	Sex
C	59.89	167
Q	13.34	76
S	27.05	642

In [72]:
```
(df1.groupby(['Survived',"Sex"])['Fare'].count()/df1.groupby(['Survived'])['Fare'].count()).to_frame()*100
```

Out[72]:

Survived	Sex	Fare
0	female	14.81
	male	85.19
1	female	67.94
	male	32.06

In [73]:
```
df1.groupby('Sex')['Embarked'].count().nlargest(2).reset_index()
```

Out[73]:

	Sex	Embarked
0	male	575
1	female	310

In [74]:
```
df1[['Pclass', 'Fare']].groupby(['Pclass'], as_index=True).mean()
```

Out[74]:

Pclass	Fare
1	84.36
2	20.66
3	13.67

In [75]:
```
#pivot_table

quality_pivot = df1.pivot_table(index='Pclass',values='Age', aggfunc=np.mean)
quality_pivot
```

Out[75]:

Pclass	Age
1	38.25
2	29.88
3	25.21

In [76]:
```
x=pd.DataFrame(pd.pivot_table(df1,index=['Sex','Embarked'],aggfunc='count')['Fare'])
x
```

Out[76]:

		Fare
Sex	**Embarked**	
female	**C**	72
	Q	36
	S	202
male	**C**	95
	Q	40
	S	440

In [77]:
```
quality_pivot = df1.pivot_table(index='Pclass',values='Age', aggfunc=np.median)
quality_pivot
```

Out[77]:

	Age
Pclass	
1	37.0
2	29.0
3	24.0

In [78]:
```
pd.crosstab(df1['Pclass'],df1['Survived'])
```

Out[78]:

Survived	**0**	**1**
Pclass		
1	80	134
2	97	87
3	370	119

In [79]:
```
#Cross Tab
pd.crosstab(df1['Sex'],df1['Embarked'])
```

Out[79]:

Embarked	**C**	**Q**	**S**
Sex			
female	72	36	202
male	95	40	440

In [80]:
```
#Cross Tab
plot_criteria= ['Sex', 'Pclass']
cm = sns.light_palette("red", as_cmap=True)
(round(pd.crosstab(df1[plot_criteria[0]], df1[plot_criteria[1]], normalize='columns') * 100,2)).style.backgroun
```

Out[80]:

Pclass	**1**	**2**	**3**
Sex			
female	42.990000	41.300000	29.450000
male	57.010000	58.700000	70.550000

In [81]:
```
pd.crosstab(df1['Sex'],df1['Embarked'],normalize = "index" ).style.background_gradient(cmap='crest')
```

Out[81]:

Embarked	**C**	**Q**	**S**
Sex			
female	0.232258	0.116129	0.651613
male	0.165217	0.069565	0.765217

In [82]:
```
plot_criteria= ['Embarked', 'Pclass']
cm = sns.light_palette("red", as_cmap=True)
(round(pd.crosstab(df1[plot_criteria[0]], df1[plot_criteria[1]], normalize='columns') * 100,2)).style.backgroun
```

Out[82]:

Pclass	**1**	**2**	**3**
Embarked			
C	39.620000	9.240000	13.500000
Q	0.940000	1.630000	14.520000
S	59.430000	89.130000	71.980000

In [83]:
```
pd.crosstab(df1['Pclass'], df1['Survived'], margins=True)
```

Out[83]:

Survived	0	1	All
Pclass			
1	80	134	214
2	97	87	184
3	370	119	489
All	547	340	887

In [84]:
```
#Create New Column

df1['sex Titanic map']=df1['Sex'].map({'male':1,'female':0})
df1.head()
```

Out[84]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0

In [85]:
```
df1["Fare Range"] = np.where(df1["Fare"] < 200, "low", "high")
df1.head()
```

Out[85]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1	low
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1	low
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low

In [86]:
```
df1["age_bins"]= pd.cut(df1["Age"] ,bins=[1,18,29 , 40 , 50 , 60 ,  80] ,labels=["child","teen","adult" , "fort
df1.head()
```

Out[86]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1	low	old
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen

In [87]:
```
df1['is_train'] = np.random.uniform(0,1,len(df1)) <=.75
df1.head()
```

Out[87]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_tr
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	Fa
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	T
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	T
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	1	low	old	Fa
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	T

In [88]:
```
#iloc & loc
df1.iloc[9:12, 2:5]  # [Row , Column]
```

Out[88]:

	Pclass	Name	Sex
13	3	Andersson, Mr. Anders Johan	male
14	3	Vestrom, Miss. Hulda Amanda Adolfina	female
15	2	Hewlett, Mrs. (Mary D Kingcome)	female

In [89]:
```
df1.iloc[2:4, 3:6]
```

Out[89]:

	Name	Sex	Age
4	Allen, Mr. William Henry	male	35.0
6	McCarthy, Mr. Timothy J	male	54.0

In [90]:
```
df1.iloc[0:4, 3] = "Anonymous"
df1.head()
```

Out[90]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
0	1	0	3	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	F
2	3	1	3	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	
4	5	0	3	Anonymous	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	
6	7	0	1	Anonymous	male	54.0	0	0	17463	51.86	E46	S	1	low	old	F
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	

In [91]:
```
df1.iloc[[3,6,9],[2,3]]
```

Out[91]:

	Pclass	Name
6	1	Anonymous
10	3	Sandstrom, Miss. Marguerite Rut
13	3	Andersson, Mr. Anders Johan

In [92]:
```
#Replace
df1["Survived"].replace({0:"Died" , 1:"Saved"} , inplace=True)
df1.head()
```

Out[92]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
0	1	Died	3	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	
2	3	Saved	3	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	
4	5	Died	3	Anonymous	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	
6	7	Died	1	Anonymous	male	54.0	0	0	17463	51.86	E46	S	1	low	old	
8	9	Saved	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	

In [93]:
```
#Rename
df1.rename(columns={"Name" : 'Person Name'},inplace=True)
df1.head()
```

Out[93]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is_
0	1	Died	3	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	low	teen	
2	3	Saved	3	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0	low	teen	
4	5	Died	3	Anonymous	male	35.0	0	0	373450	8.05	NaN	S	1	low	adult	
6	7	Died	1	Anonymous	male	54.0	0	0	17463	51.86	E46	S	1	low	old	
8	9	Saved	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0	low	teen	

In [94]:
```
# Replace the codes with their full names
df1['Embarked'] = df1['Embarked'].replace({'S': 'Southampton', 'C': 'Cherbourg', 'Q': 'Queenstown'})
df1.sample(5)
```

Out[94]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	is
290	291	Saved	1	Barber, Miss. Ellen "Nellie"	female	26.0	0	0	19877	78.85	NaN	Southampton	0	low	teen	
887	888	Saved	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	Southampton	0	low	teen	
135	136	Died	2	Richard, Mr. Emile	male	23.0	0	0	SC/PARIS 2133	15.05	NaN	Cherbourg	1	low	teen	
41	42	Died	2	Turpin, Mrs. William John Robert (Dorothy Ann ...	female	27.0	1	0	11668	21.00	NaN	Southampton	0	low	teen	
500	501	Died	3	Calic, Mr. Petar	male	17.0	0	0	315086	8.66	NaN	Southampton	1	low	child	

In [95]:
```
df1['Pclass'][df1['Pclass'] == 1] = 'Rich'
df1['Pclass'][df1['Pclass'] == 2] = 'Middel Class'
df1['Pclass'][df1['Pclass'] == 3] = 'Poor'
df1.head()
```

Out[95]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	i
0	1	Died	Poor	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	Southampton	1	low	teen	
2	3	Saved	Poor	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	Southampton	0	low	teen	
4	5	Died	Poor	Anonymous	male	35.0	0	0	373450	8.05	NaN	Southampton	1	low	adult	
6	7	Died	Rich	Anonymous	male	54.0	0	0	17463	51.86	E46	Southampton	1	low	old	
8	9	Saved	Poor	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	Southampton	0	low	teen	

In [96]:
```
#Converting Zero Value to NaN Value
df1.loc[df1['Fare'] == 0,'Fare'] = np.nan
df1.head()
```

Out[96]:

	PassengerId	Survived	Pclass	Person Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	sex Titanic map	Fare Range	age_bins	i
0	1	Died	Poor	Anonymous	male	22.0	1	0	A/5 21171	7.25	NaN	Southampton	1	low	teen	
2	3	Saved	Poor	Anonymous	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	Southampton	0	low	teen	
4	5	Died	Poor	Anonymous	male	35.0	0	0	373450	8.05	NaN	Southampton	1	low	adult	
6	7	Died	Rich	Anonymous	male	54.0	0	0	17463	51.86	E46	Southampton	1	low	old	
8	9	Saved	Poor	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	Southampton	0	low	teen	

In [97]:
```
# Replace First Three name with any symbol or Value
df1 = df.copy()
df1.loc[0:2, 'Name'] = '?'
df1.head()
```

Out[97]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	?	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	?	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S

In [98]:
```
#Replace '?' values in workclass variable with NaN
df1['Name'].replace('?', np.NaN, inplace=True)
df1.head()
```

Out[98]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	NaN	male	22.0	1	0	A/5 21171	7.25	NaN	S
2	3	1	3	NaN	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S

In [99]:
```
#Saving to Excel Format
df1.to_excel('titanic.xlsx',sheet_name="Passenger",index=False)
```

In [100…
```
df1 = df.copy()
df1[df1['Age'].isnull()].index
```

Out[100]:
```
Index([ 17,  19,  26,  28,  29,  31,  32,  36,  42,  45,
       ...
       832, 837, 839, 846, 849, 859, 863, 868, 878, 888],
      dtype='int64', length=176)
```

In [101…
```
#Join
```

In [101…
```
#Join
Join = df1.join(df1, lsuffix = '_1') # lsuffix = Left Suffix
Join.head(2)
```

Out[101]:

	PassengerId_1	Survived_1	Pclass_1	Name_1	Sex_1	Age_1	SibSp_1	Parch_1	Ticket_1	Fare_1	Cabin_1	Embarked_1	PassengerI
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	

In [102…
```
#Melt
Melt = pd.melt(df1,id_vars = ['Embarked'], value_vars = ['Survived'])
Melt.head()
```

Out[102]:

	Embarked	variable	value
0	S	Survived	0
1	S	Survived	1
2	S	Survived	0
3	S	Survived	0
4	S	Survived	1

In [103…
```
df1 = df.copy()
df1.dtypes
```

Out[103]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [104…
```
df1["Sex"] = df1.Sex.apply(lambda  x:'male' if x==1 else 'female')
```

In [105…
```
df1['Pclass_New'] = df1['Pclass'].apply(lambda x: 'UpperClass' if x == 1 else 0)
display(df1.head())
df1['Pclass_New'].value_counts().to_frame()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Pclass_New
0	1	0	3	Braund, Mr. Owen Harris	female	22.0	1	0	A/5 21171	7.25	NaN	S	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	0
4	5	0	3	Allen, Mr. William Henry	female	35.0	0	0	373450	8.05	NaN	S	0
6	7	0	1	McCarthy, Mr. Timothy J	female	54.0	0	0	17463	51.86	E46	S	UpperClass
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	0

Out[105]:

Pclass_New	count
0	673
UpperClass	214

In [106…
```
df1['Pclass_New'] = df1['Pclass'].apply(lambda x: 5 if x > 2 else 0)
display(df1.head())
df1['Pclass_New'].value_counts().to_frame()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Pclass_New
0	1	0	3	Braund, Mr. Owen Harris	female	22.0	1	0	A/5 21171	7.25	NaN	S	5
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	5
4	5	0	3	Allen, Mr. William Henry	female	35.0	0	0	373450	8.05	NaN	S	5
6	7	0	1	McCarthy, Mr. Timothy J	female	54.0	0	0	17463	51.86	E46	S	0
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	5

Out[106]:

Pclass_New	count
5	489
0	398

In [107…
```
df1.columns.tolist()
```

Out[107]:
```
['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked',
 'Pclass_New']
```

In [108…
```
df1.nunique()
```

Out[108]:
```
PassengerId    887
Survived         2
Pclass           3
Name           887
Sex              1
Age             88
SibSp            7
Parch            7
Ticket         679
Fare           246
Cabin          146
Embarked         3
Pclass_New       2
dtype: int64
```

In [109…
```
df1 = df.copy()

women = df1.loc[df1['Sex'] == 'female']["Survived"]
rate_women = (women.sum()/len(women)).round(3)*100
print("Percentage of women who survived:", rate_women,"%")

men = df1.loc[df1['Sex'] == 'male']["Survived"]
rate_men = (men.sum()/len(men)).round(3)*100
print("Percentage of men who survived  :", rate_men,"%")
```

```
Percentage of women who survived: 74.0 %
Percentage of men who survived  : 19.0 %
```

In [110…
```
df1 = df.copy()
df1['Age_Range'] = pd.cut(df1['Age'],
bins=[0.,15,30,45,60,65,np.inf],
labels=[1,2,3,4,5,6])
df1.head()
```

Out[110]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_Range
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	3
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	4
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	2

In [111…
```
df1 = df.copy()
df1['AgeBand'] = pd.cut(df1['Age'], 5)
```

```
df1[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='AgeBand', ascending=Tr
```

Out[111]:

	AgeBand	Survived
0	(0.34, 16.336]	0.56
1	(16.336, 32.252]	0.37
2	(32.252, 48.168]	0.40
3	(48.168, 64.084]	0.43
4	(64.084, 80.0]	0.09

In [112…
```
Cabin_Not_NA = df1[df1['Cabin'].notna()]
Cabin_Not_NA.head()
```

Out[112]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	AgeBand
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	(48.168, 64.084]
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.70	G6	S	(0.34, 16.336]
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.55	C103	S	(48.168, 64.084]
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.00	D56	S	(32.252, 48.168]
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.50	A6	S	(16.336, 32.252]

In [113…
```
Cabin_NaN = df1[df1['Cabin'].isnull()]
Cabin_NaN.head()
```

Out[113]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	AgeBand
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	(16.336, 32.252]
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	(16.336, 32.252]
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	(32.252, 48.168]
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	(16.336, 32.252]
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.07	NaN	C	(0.34, 16.336]

In [114…
```
df1 = pd.get_dummies(df1, columns = ['Embarked'], drop_first=True)
df1.head()
```

Out[114]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	AgeBand	Embarked_Q	Embarked_S
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	(16.336, 32.252]	False	True
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	(16.336, 32.252]	False	True
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	(32.252, 48.168]	False	True
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	(48.168, 64.084]	False	True
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	(16.336, 32.252]	False	True

In [115…
```
df1 = df.copy()

def Grade(Percentage):
    if Percentage >= 500:
        return 'High'
    if Percentage >= 300:
        return 'Medium'
    if Percentage >= 200:
        return 'Average'
    if Percentage >= 100:
        return 'Low'
    if Percentage >= 50:
```

```
        return 'VeryLow'
    return 'Free'

df1['Fare_Range']=df1.apply(lambda x: Grade(x['Fare']),axis=1)
df1.head()
```

Out[115]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_Range
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	Free
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	NaN	S	Free
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S	Free
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.86	E46	S	VeryLow
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.13	NaN	S	Free

In [116…
```
wine = pd.read_csv('WineQT.csv')
wine.head(3)
```

Out[116]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.09	15.0	54.0	1.0	3.26	0.65	9.8	5	2

In [117…
```
# Create correlation matrix
corr_matrix = wine.corr().abs()
# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
# Find features with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.30)]
to_drop
```

Out[117]:
```
['citric acid',
 'total sulfur dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol',
 'quality',
 'Id']
```

In [118…
```
wine['good_quality']=["yes" if x>=7 else 'no' for x in wine['quality']]
wine.head()
```

Out[118]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id	good_quality
0	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0	no
1	7.8	0.88	0.00	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1	no
2	7.8	0.76	0.04	2.3	0.09	15.0	54.0	1.0	3.26	0.65	9.8	5	2	no
3	11.2	0.28	0.56	1.9	0.07	17.0	60.0	1.0	3.16	0.58	9.8	6	3	no
4	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	4	no

In [119…
```
#Removing all Negative values

wine['residual sugar'] = wine['residual sugar'].abs()
wine.head()
```

Out[119]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id	good_quality
0	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	0	no
1	7.8	0.88	0.00	2.6	0.10	25.0	67.0	1.0	3.20	0.68	9.8	5	1	no
2	7.8	0.76	0.04	2.3	0.09	15.0	54.0	1.0	3.26	0.65	9.8	5	2	no
3	11.2	0.28	0.56	1.9	0.07	17.0	60.0	1.0	3.16	0.58	9.8	6	3	no
4	7.4	0.70	0.00	1.9	0.08	11.0	34.0	1.0	3.51	0.56	9.4	5	4	no

In [120…
```
# Print Row
row_30 = wine.iloc[75]
print(row_30)
```

```
fixed acidity            7.8
volatile acidity        0.41
citric acid             0.68
residual sugar           1.7
chlorides               0.47
free sulfur dioxide     18.0
total sulfur dioxide    69.0
density                  1.0
pH                      3.08
sulphates               1.31
alcohol                  9.3
quality                    5
Id                       106
good_quality              no
Name: 75, dtype: object
```

In [121…
```
any_negative_yield = (wine['chlorides'] < 0).any()

if any_negative_yield:
    print("The 'chlorides' column contains negative values.")
else:
    print("The 'chlorides' column does not contain negative values.")
```

```
The 'chlorides' column does not contain negative values.
```

In [122…
```
geo = pd.read_csv('gapminder_full.csv')
geo.head(3)
```

Out[122]:

	country	year	population	continent	life_exp	gdp_cap
0	Afghanistan	1952	8425333	Asia	28.80	779.45
1	Afghanistan	1957	9240934	Asia	30.33	820.85
2	Afghanistan	1962	10267083	Asia	32.00	853.10

In [123…
```
geo[(geo['population']>10000000) & (geo['country']=='Afghanistan')][['gdp_cap','life_exp','continent']]
```

Out[123]:

	gdp_cap	life_exp	continent
2	853.10	32.00	Asia
3	836.20	34.02	Asia
4	739.98	36.09	Asia
5	786.11	38.44	Asia
6	978.01	39.85	Asia
7	852.40	40.82	Asia
8	649.34	41.67	Asia
9	635.34	41.76	Asia
10	726.73	42.13	Asia
11	974.58	43.83	Asia

In [124…
```
reg_medal=geo.groupby(['country','continent']).size().reset_index().head(10)
reg_medal
```

Out[124]:

	country	continent	0
0	Afghanistan	Asia	12
1	Albania	Europe	12
2	Algeria	Africa	12
3	Angola	Africa	12
4	Argentina	Americas	12
5	Australia	Oceania	12
6	Austria	Europe	12
7	Bahrain	Asia	12
8	Bangladesh	Asia	12
9	Belgium	Europe	12

In [125…
```
df2=geo.groupby('country')['continent'].nunique().reset_index()
df2.head()
```

Out[125]:

	country	continent
0	Afghanistan	1
1	Albania	1
2	Algeria	1
3	Angola	1
4	Argentina	1

In [126…
```
geo.groupby('country')['continent'].count().nlargest(20).reset_index().head(10)
```

Out[126]:

	country	continent
0	Afghanistan	12
1	Albania	12
2	Algeria	12
3	Angola	12
4	Argentina	12
5	Australia	12
6	Austria	12
7	Bahrain	12
8	Bangladesh	12
9	Belgium	12

In [127…
```
Highest_Population = geo.nlargest(10, 'population', keep='all')
Highest_Population.head(10)
```

Out[127]:

	country	year	population	continent	life_exp	gdp_cap
299	China	2007	1318683096	Asia	72.96	4959.11
298	China	2002	1280400000	Asia	72.03	3119.28
297	China	1997	1230075000	Asia	70.43	2289.23
296	China	1992	1164970000	Asia	68.69	1655.78
707	India	2007	1110396331	Asia	64.70	2452.21
295	China	1987	1084035000	Asia	67.27	1378.90
706	India	2002	1034172547	Asia	62.88	1746.77
294	China	1982	1000281000	Asia	65.53	962.42
705	India	1997	959000000	Asia	61.77	1458.82
293	China	1977	943455000	Asia	63.97	741.24

In [128…
```
Lowest_Population = geo.nsmallest(10, 'population', keep='all')
Lowest_Population.head(10)
```

Out[128]:

	country	year	population	continent	life_exp	gdp_cap
1296	Sao Tome and Principe	1952	60011	Africa	46.47	879.58
1297	Sao Tome and Principe	1957	61325	Africa	48.95	860.74
420	Djibouti	1952	63149	Africa	34.81	2669.53
1298	Sao Tome and Principe	1962	65345	Africa	51.89	1071.55
1299	Sao Tome and Principe	1967	70787	Africa	54.42	1384.84
421	Djibouti	1957	71851	Africa	37.33	2864.97
1300	Sao Tome and Principe	1972	76595	Africa	56.48	1532.99
1301	Sao Tome and Principe	1977	86796	Africa	58.55	1737.56
422	Djibouti	1962	89898	Africa	39.69	3020.99
1302	Sao Tome and Principe	1982	98593	Africa	60.35	1890.22

In [129…
```
Population = geo.groupby(by = ['country'], axis = 0)['population'].sum()
Range = pd.DataFrame(Population).reset_index()
Range.sort_values(by= ['population'], ascending = False, inplace = True)
Range.head(10)
```

Out[129]:

	country	population
24	China	11497920623
58	India	8413568878
134	United States	2738534790
59	Indonesia	1779874000
14	Brazil	1467745520
66	Japan	1341105696
97	Pakistan	1124200629
8	Bangladesh	1089064744
47	Germany	930564520
94	Nigeria	884496214

In [130…
```
print(f"\033[031m\033[1m")
print("Unique continent Names :", geo['continent'].nunique())
geo['continent'].value_counts().nlargest(10).to_frame().style.background_gradient(cmap='copper')
```

Unique continent Names : 5

Out[130]:

continent	count
Africa	624
Asia	396
Europe	360
Americas	300
Oceania	24

In [131…
```
df1 = geo[geo['continent']=='Asia']
df1.groupby('country')['life_exp'].max().sort_values(ascending=False).head(5).reset_index()
```

Out[131]:

	country	life_exp
0	Japan	82.60
1	Hong Kong, China	82.21
2	Israel	80.75
3	Singapore	79.97
4	Korea, Rep.	78.62

In [132…
```
geo.groupby('country')['gdp_cap'].mean().sort_values(ascending=False).index[0:5]
```

Out[132]:
```
Index(['Kuwait', 'Switzerland', 'Norway', 'United States', 'Canada'], dtype='object', name='country')
```

In [133…
```
geo.groupby('country')['gdp_cap'].mean().sort_values(ascending=False).head(10)
```

Out[133]:
```
country
Kuwait           65332.91
Switzerland      27074.33
Norway           26747.31
United States    26261.15
Canada           22410.75
Netherlands      21748.85
Denmark          21671.82
Germany          20556.68
Iceland          20531.42
Austria          20411.92
Name: gdp_cap, dtype: float64
```

In [134…
```
data = geo[geo['continent']=='Africa']
data[data['gdp_cap'] == data['gdp_cap'].max()]['country']
```

Out[134]:
```
905    Libya
Name: country, dtype: object
```

In [135…
```
data = geo[geo['continent']=='Africa']
data[data['gdp_cap'] == data['gdp_cap'].min()]['country']
```

Out[135]:
```
334    Congo, Dem. Rep.
Name: country, dtype: object
```

In [136…
```
df1 = geo[(geo['continent'] == 'Europe')]
df1 = df1[df1['gdp_cap'] == df1['gdp_cap'].max()]
df1
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from lazypredict.Supervised import LazyClassifier
import matplotlib.pyplot as plt

# Assuming df is your DataFrame containing the Titanic dataset
titanic_data = df.copy()
titanic_data = titanic_data.dropna()

# Select relevant features
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']
X = titanic_data[features]
y = titanic_data['Survived']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
andom_state=42)

# Create a LazyClassifier
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metri
c=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
print("Models performance:")
models
```

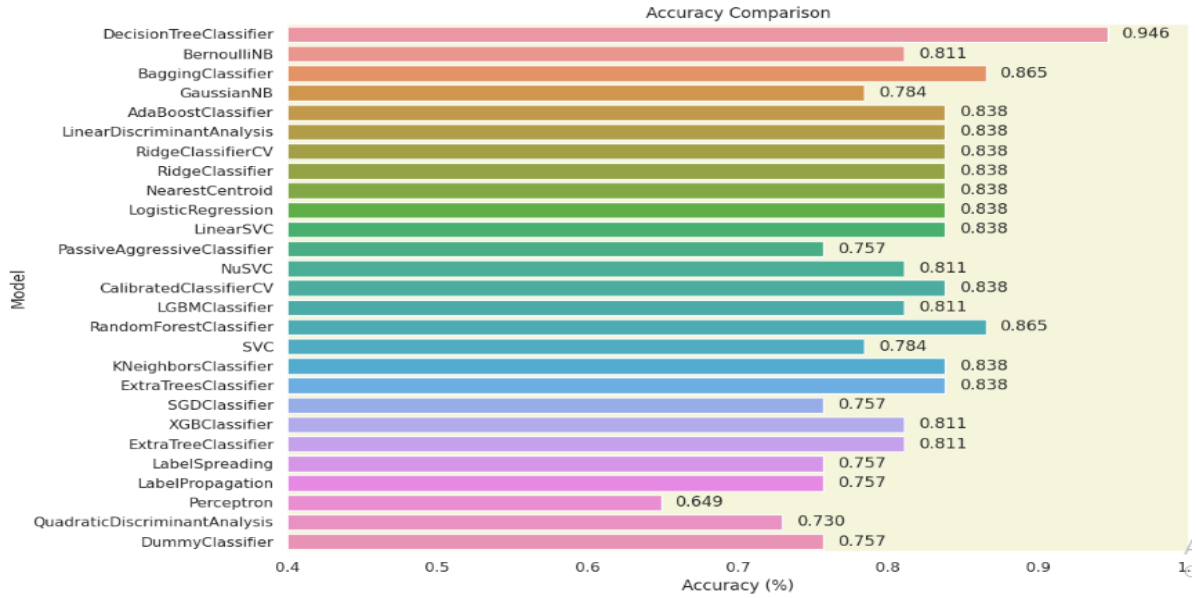
Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
DecisionTreeClassifier	0.95	0.93	0.93	0.95	0.02
BernoulliNB	0.81	0.84	0.84	0.82	0.03
BaggingClassifier	0.86	0.84	0.84	0.87	0.06
GaussianNB	0.78	0.82	0.82	0.80	0.02
AdaBoostClassifier	0.84	0.82	0.82	0.84	0.14
LinearDiscriminantAnalysis	0.84	0.82	0.82	0.84	0.04
RidgeClassifierCV	0.84	0.82	0.82	0.84	0.02
RidgeClassifier	0.84	0.82	0.82	0.84	0.03
NearestCentroid	0.84	0.82	0.82	0.84	0.03
LogisticRegression	0.84	0.82	0.82	0.84	0.03
LinearSVC	0.84	0.82	0.82	0.84	0.03
PassiveAggressiveClassifier	0.76	0.80	0.80	0.77	0.03
NuSVC	0.81	0.80	0.80	0.82	0.13
CalibratedClassifierCV	0.84	0.78	0.78	0.84	0.07

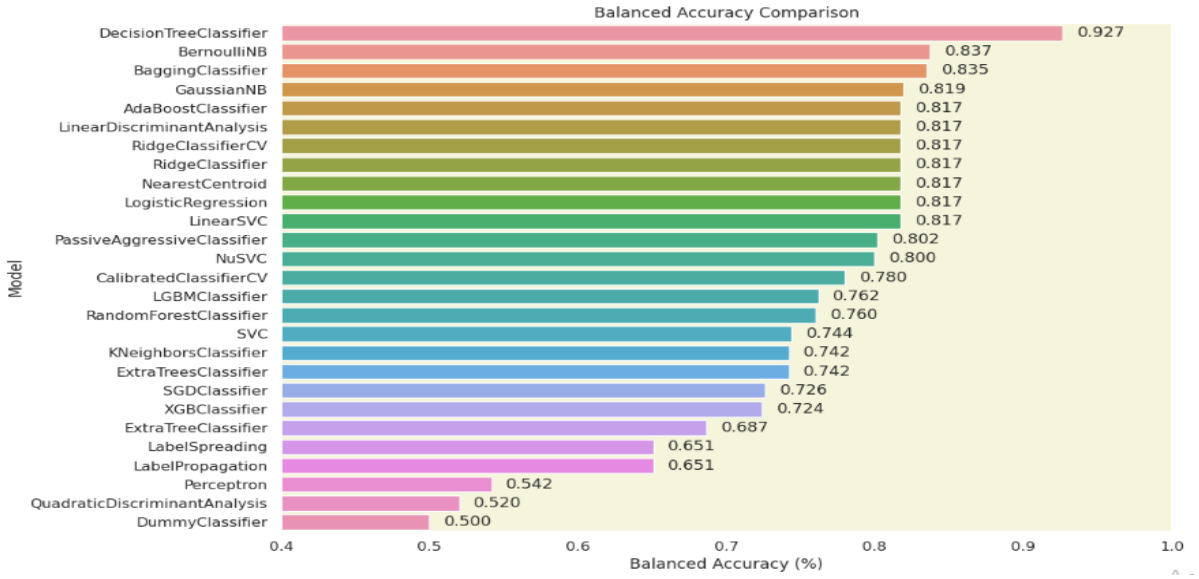
LGBMClassifier	0.81	0.76	0.76	0.81	0.08
RandomForestClassifier	0.86	0.76	0.76	0.85	0.25
SVC	0.78	0.74	0.74	0.79	0.02
KNeighborsClassifier	0.84	0.74	0.74	0.83	0.03
ExtraTreesClassifier	0.84	0.74	0.74	0.83	0.20
SGDClassifier	0.76	0.73	0.73	0.77	0.02
XGBClassifier	0.81	0.72	0.72	0.81	0.11
ExtraTreeClassifier	0.81	0.69	0.69	0.80	0.02
LabelSpreading	0.76	0.65	0.65	0.75	0.02
LabelPropagation	0.76	0.65	0.65	0.75	0.02
Perceptron	0.65	0.54	0.54	0.65	0.02
QuadraticDiscriminantAnalysis	0.73	0.52	0.52	0.68	0.03
DummyClassifier	0.76	0.50	0.50	0.65	0.02

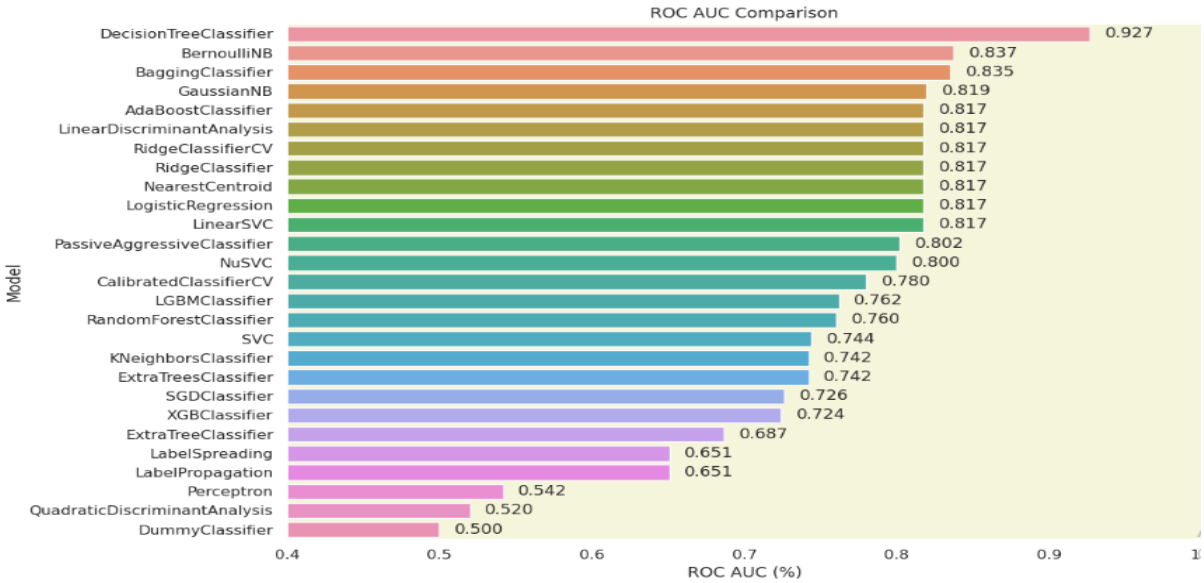
```
# If you want to evaluate a specific model (e.g., the best performing one)
if not models.empty:
    best_model = models.index[0]
    print(f"\nBest model: {best_model}")

    if best_model in predictions.columns:
        best_model_predictions = predictions[best_model]
        accuracy = accuracy_score(y_test, best_model_predictions)
        print(f"Accuracy of the best model: {accuracy}")
    else:
        print(f"Warning: Predictions for {best_model} not found in the pr
edictions DataFrame.")
        print("Available models in predictions:")
        print(predictions.columns)
else:
    print("No models were successfully trained.")

print("\nShape of predictions DataFrame:", predictions.shape)
print("\nFirst few rows of predictions DataFrame:")
predictions
```

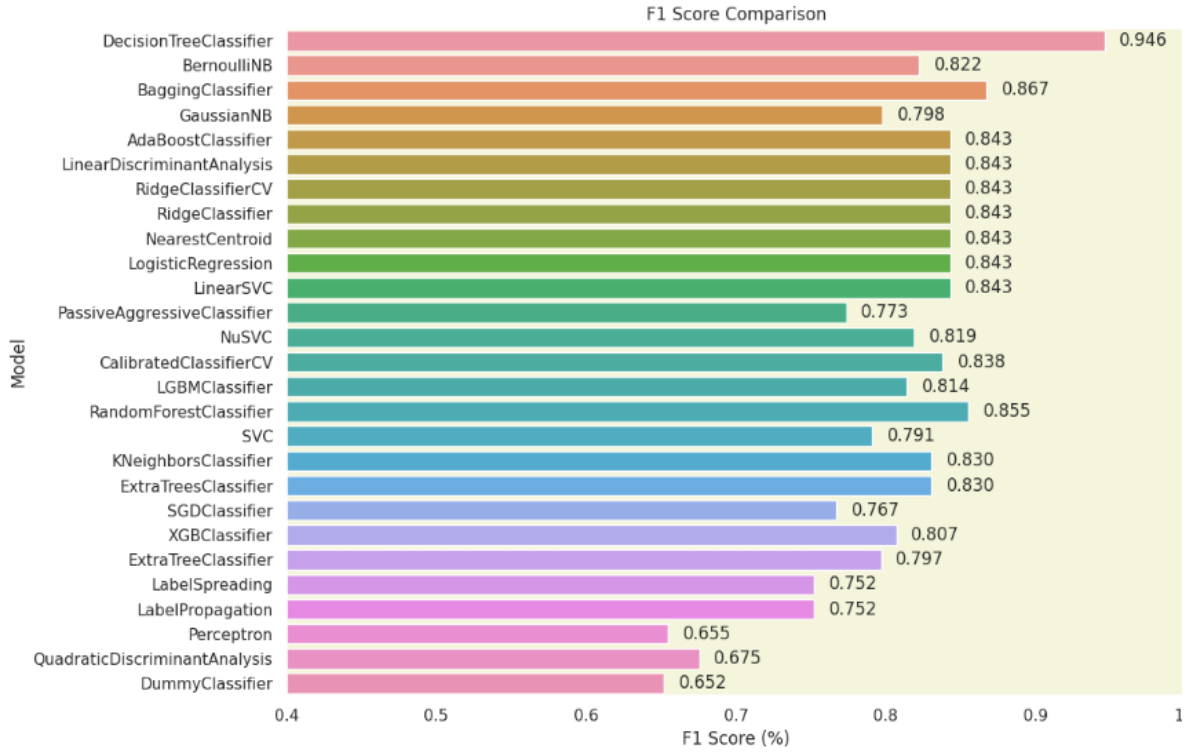
Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
DecisionTreeClassifier	0.95	0.93	0.93	0.95	0.02
BernoulliNB	0.81	0.84	0.84	0.82	0.03
BaggingClassifier	0.86	0.84	0.84	0.87	0.06
GaussianNB	0.78	0.82	0.82	0.80	0.02
AdaBoostClassifier	0.84	0.82	0.82	0.84	0.14
LinearDiscriminantAnalysis	0.84	0.82	0.82	0.84	0.04
RidgeClassifierCV	0.84	0.82	0.82	0.84	0.02
RidgeClassifier	0.84	0.82	0.82	0.84	0.03
NearestCentroid	0.84	0.82	0.82	0.84	0.03
LogisticRegression	0.84	0.82	0.82	0.84	0.03
LinearSVC	0.84	0.82	0.82	0.84	0.03
PassiveAggressiveClassifier	0.76	0.80	0.80	0.77	0.03
NuSVC	0.81	0.80	0.80	0.82	0.13
CalibratedClassifierCV	0.84	0.78	0.78	0.84	0.07
LGBMClassifier	0.81	0.76	0.76	0.81	0.08
RandomForestClassifier	0.86	0.76	0.76	0.85	0.25
SVC	0.78	0.74	0.74	0.79	0.02
KNeighborsClassifier	0.84	0.74	0.74	0.83	0.03
ExtraTreesClassifier	0.84	0.74	0.74	0.83	0.20
SGDClassifier	0.76	0.73	0.73	0.77	0.02

```
plt.figure(figsize=(12, 30))
metrics = ['Accuracy', 'Balanced Accuracy', 'ROC AUC', 'F1 Score']

for i, metric in enumerate(metrics):
    plt.subplot(4, 1, i+1)
    ax = sns.barplot(x=models[metric], y=models.index, orient='h')
    plt.title(f'{metric} Comparison')
    plt.xlabel(f'{metric} (%)')
    plt.ylabel('Model')

    # Set x-axis to start from 0.40
    plt.xlim(0.40, 1.0)

    # Add value labels on the bars
    for j, v in enumerate(models[metric]):
        ax.text(v + 0.01, j, f'{v:.3f}', va='center')

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

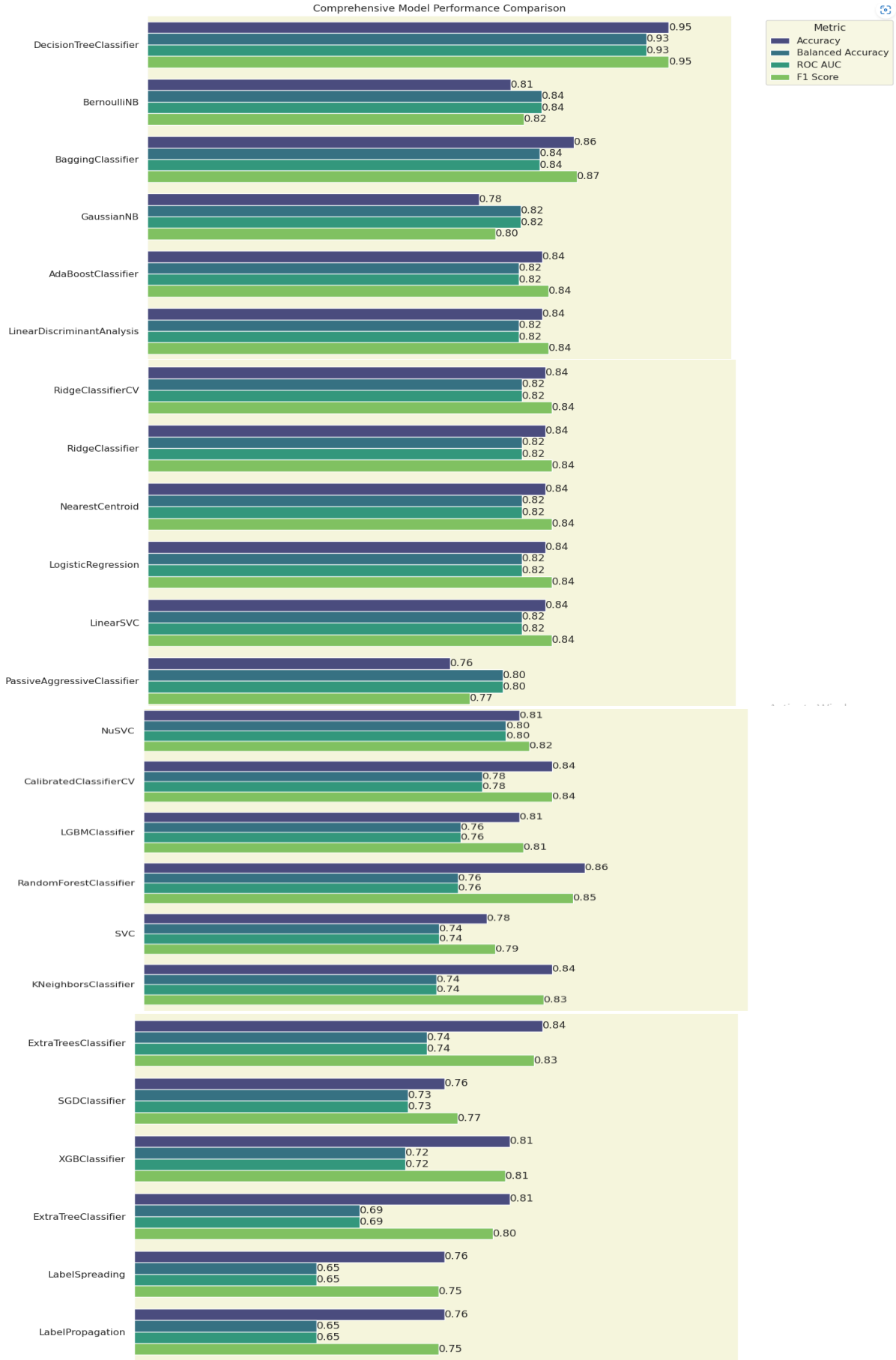

Accuracy Comparison
DecisionTreeClassifier 0.946
BernoulliNB 0.811
BaggingClassifier 0.865
GaussianNB 0.784
AdaBoostClassifier 0.838
LinearDiscriminantAnalysis 0.838
RidgeClassifierCV 0.838
RidgeClassifier 0.838
NearestCentroid 0.838
LogisticRegression 0.838
LinearSVC 0.838
PassiveAggressiveClassifier 0.757
NuSVC 0.811
CalibratedClassifierCV 0.838
LGBMClassifier 0.811
RandomForestClassifier 0.865
SVC 0.784
KNeighborsClassifier 0.838
ExtraTreesClassifier 0.838
SGDClassifier 0.757
XGBClassifier 0.811
ExtraTreeClassifier 0.811
LabelSpreading 0.757
LabelPropagation 0.757
Perceptron 0.649
QuadraticDiscriminantAnalysis 0.730
DummyClassifier 0.757
Model
0.4 0.5 0.6 0.7 0.8 0.9
Accuracy (%)


Balanced Accuracy Comparison
DecisionTreeClassifier 0.927
BernoulliNB 0.837
BaggingClassifier 0.835
GaussianNB 0.819
AdaBoostClassifier 0.817
LinearDiscriminantAnalysis 0.817
RidgeClassifierCV 0.817
RidgeClassifier 0.817
NearestCentroid 0.817
LogisticRegression 0.817
LinearSVC 0.817
PassiveAggressiveClassifier 0.802
NuSVC 0.800
CalibratedClassifierCV 0.780
LGBMClassifier 0.762
RandomForestClassifier 0.760
SVC 0.744
KNeighborsClassifier 0.742
ExtraTreesClassifier 0.742
SGDClassifier 0.726
XGBClassifier 0.724
ExtraTreeClassifier 0.687
LabelSpreading 0.651
LabelPropagation 0.651
Perceptron 0.542
QuadraticDiscriminantAnalysis 0.520
DummyClassifier 0.500
Model
0.4 0.5 0.6 0.7 0.8 0.9 1.0
Balanced Accuracy (%)


ROC AUC Comparison
DecisionTreeClassifier 0.927
BernoulliNB 0.837
BaggingClassifier 0.835
GaussianNB 0.819
AdaBoostClassifier 0.817
LinearDiscriminantAnalysis 0.817
RidgeClassifierCV 0.817
RidgeClassifier 0.817
NearestCentroid 0.817
LogisticRegression 0.817
LinearSVC 0.817
PassiveAggressiveClassifier 0.802
NuSVC 0.800
CalibratedClassifierCV 0.780
LGBMClassifier 0.762
RandomForestClassifier 0.760
SVC 0.744
KNeighborsClassifier 0.742
ExtraTreesClassifier 0.742
SGDClassifier 0.726
XGBClassifier 0.724
ExtraTreeClassifier 0.687
LabelSpreading 0.651
LabelPropagation 0.651
Perceptron 0.542
QuadraticDiscriminantAnalysis 0.520
DummyClassifier 0.500
Model
0.4 0.5 0.6 0.7 0.8 0.9
ROC AUC (%)




```
# Assuming 'models' and 'metrics' are already defined DataFrames
summary_df = models[metrics].copy()
summary_df['Model'] = models.index
summary_df = summary_df.melt(id_vars=['Model'], var_name='Metric'
, value_name='Score')

plt.figure(figsize=(15, 30))
barplot = sns.barplot(x='Score', y='Model', hue='Metric', data=summ
ary_df, orient='h', palette='viridis')

# Add values at the edge of the bars
for container in barplot.containers:
    barplot.bar_label(container, fmt='%.2f', label_type='edge')

plt.title('Comprehensive Model Performance Comparison')
plt.xlabel('Score (%)')
plt.ylabel('Model')
plt.legend(title='Metric', bbox_to_anchor=(1.05, 1), loc='upper left')

# Set x-axis to start from 0.50
plt.xlim(0.50, 1.0)
plt.tight_layout()
plt.show()
```


Comprehensive Model Performance Comparison
Metric
Accuracy
Balanced Accuracy
ROC AUC
F1 Score
DecisionTreeClassifier
0.95
0.93
0.93
0.95
BernoulliNB
0.81
0.84
0.84
0.82
BaggingClassifier
0.86
0.84
0.84
0.87
GaussianNB
0.78
0.82
0.82
0.80
AdaBoostClassifier
0.84
0.82
0.82
0.84
LinearDiscriminantAnalysis
0.84
0.82
0.82
0.84
RidgeClassifierCV
0.84
0.82
0.82
0.84
RidgeClassifier
0.84
0.82
0.82
0.84
NearestCentroid
0.84
0.82
0.82
0.84
LogisticRegression
0.84
0.82
0.82
0.84
LinearSVC
0.84
0.82
0.82
0.84
PassiveAggressiveClassifier
0.76
0.80
0.80
0.77
NuSVC
0.81
0.80
0.80
0.82
CalibratedClassifierCV
0.84
0.78
0.78
0.84
LGBMClassifier
0.81
0.76
0.76
0.81
RandomForestClassifier
0.86
0.76
0.76
0.85
SVC
0.78
0.74
0.74
0.79
KNeighborsClassifier
0.84
0.74
0.74
0.83
ExtraTreesClassifier
0.84
0.74
0.74
0.83
SGDClassifier
0.76
0.73
0.73
0.77
XGBClassifier
0.81
0.72
0.72
0.81
ExtraTreeClassifier
0.81
0.69
0.69
0.80
LabelSpreading
0.76
0.65
0.65
0.75
LabelPropagation
0.76
0.65
0.65
0.75