Titanic Survival Prediction | ...

Draft saved

File Edit View Run Settings Add-ons Help

Run All Code Draft Session (9m)

[2]:
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from matplotlib import pylab
import xgboost as Xgb
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

```

```
/kaggle/input/titanic-dataset/Titanic-Dataset.csv
```

[3]:
```python
df = pd.read_csv("/kaggle/input/titanic-dataset/Titanic-Dataset.csv")
df.head().style.set_properties(
    **{
        'background-color':'LightBlue',
        'color':'Black',
        'border-color':'darkblack'
    })
```

[3]:

	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
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.100000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.050000	nan	S

Shape of the dataset

[4]:
```python
print("Shape of the Dataset is:", df.shape)
```

```
Shape of the Dataset is: (891, 12)
```

Getting informatiom about dataset

[5]:
```python
df.describe()
```

[5]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

[6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Visualisation on Data Type

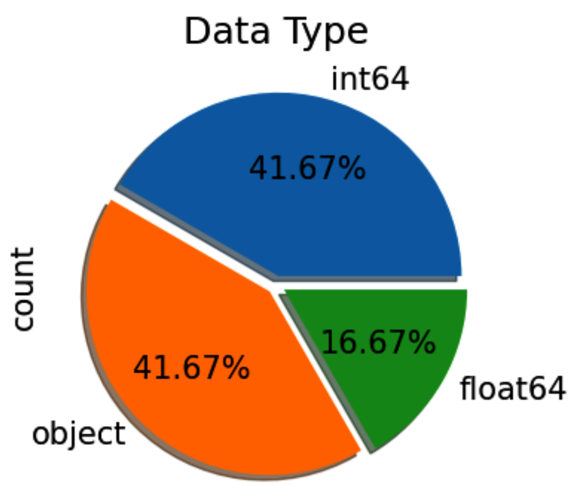
[14]:

```
plt.figure(figsize=(6,4))
plt.rcParams.update({'font.size':15})

df.dtypes.value_counts().plot.pie(explode=[0.05,0.05,0.05],
                                  autopct='%1.2f%%',
                                  shadow=True)

plt.title('Data Type',
          color='Black',
          loc='center')
```

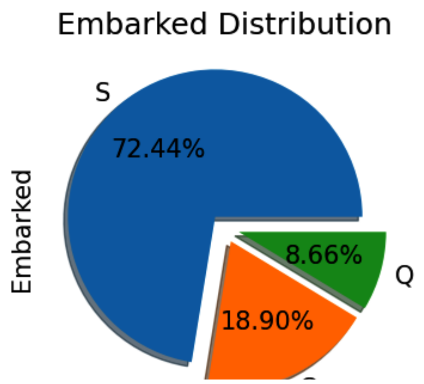
[14… Text(0.5, 1.0, 'Data Type')



Visualisation on "Embarked" column

[13]:
```
plt.figure(figsize=(6,4))
plt.rcParams.update({'font.size': 15})
df['Embarked'].value_counts().plot.pie(explode=[0.1, 0.1, 0.1],
                                        autopct='%1.2f%%',
                                        shadow=True)
plt.title('Embarked Distribution',color='Black',loc='center')
plt.ylabel('Embarked')
```

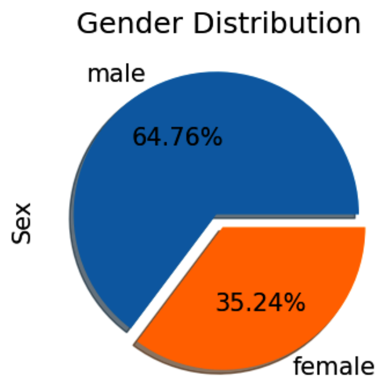
[13…
```
Text(0, 0.5, 'Embarked')
```



Visualisation on Gender Distribution

[15]:
```
plt.figure(figsize=(6,4))
plt.rcParams.update({'font.size': 15})
df['Sex'].value_counts().plot.pie(explode=[.05,.05],
                                   autopct='%1.2f%%',
                                   shadow=True)
plt.title('Gender Distribution')
plt.ylabel('Sex')
```

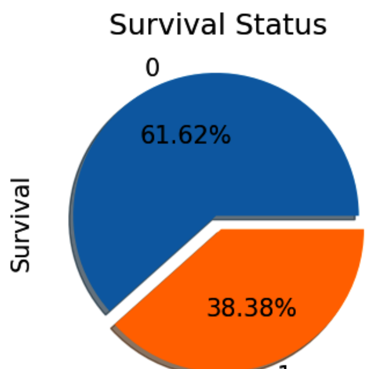
[15…
```
Text(0, 0.5, 'Sex')
```



Visualisation on Survival Status

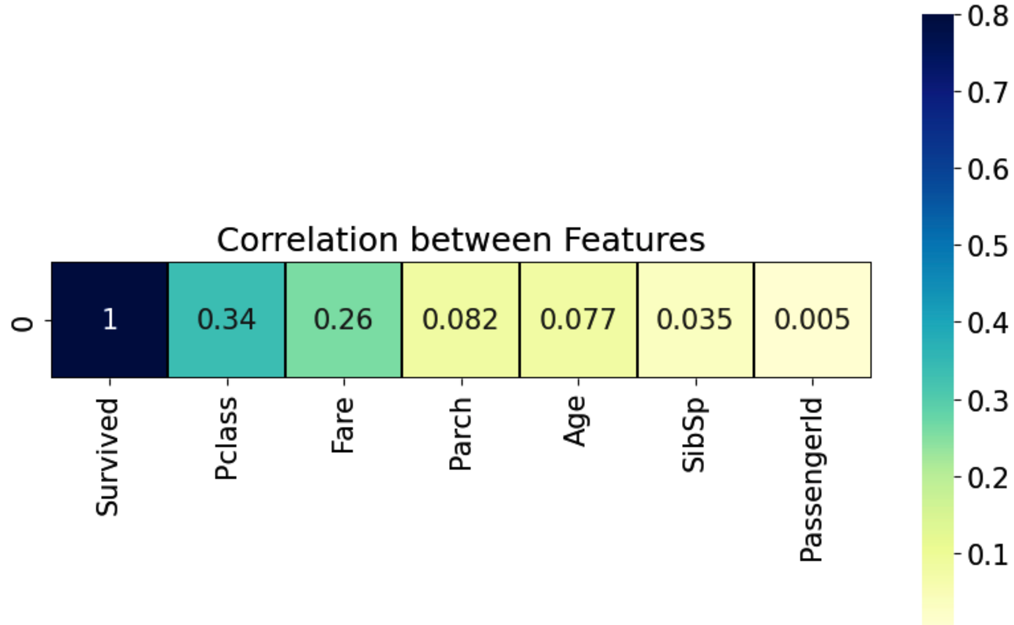
[16]:
```
plt.figure(figsize=(6,4))
plt.rcParams.update({'font.size': 15})
df['Survived'].value_counts().plot.pie(explode=[.05,.05],
                                        autopct='%1.2f%%',
                                        shadow=True)
plt.title('Survival Status', loc='center')
plt.ylabel('Survival')
print('0 = Died\n''1 = Survived')
```

```
0 = Died
1 = Survived
```



Visalisation on Correlation

[17]:
```python
numeric_df = df.select_dtypes(include=['number'])
corr = numeric_df.corrwith(numeric_df['Survived']).abs().sort_values(ascending=False)
plt.figure(figsize=(10,6))
sns.heatmap(
    corr.values.reshape(1, -1),
    vmax=0.8, linewidths=.01,
    square=True, annot=True, cmap='YlGnBu',
    linecolor="black", xticklabels=corr.index)
plt.title('Correlation between Features')
plt.show()
```



Seeing the missing values

[18]:
```python
def missing_value (df):
    missing_Number = df.isnull().sum().sort_values(ascending=False)[df.isnull().sum().sort_values(ascending=False) !=0]
    missing_percent=round((df.isnull().sum()/df.isnull().count())*100,2)[round((df.isnull().sum()/df.isnull().count())*100,
    missing = pd.concat([missing_Number,missing_percent],axis=1,keys=['Missing Number','Missing Percentage'])
    return missing
```

[19]:
```python
missing_value(df).style.background_gradient(cmap='coolwarm').format(precision=2)
```

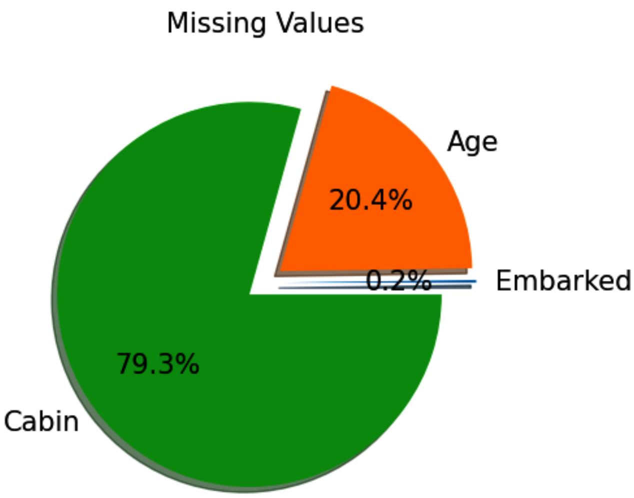
[19…

	Missing Number	Missing Percentage
Cabin	687	77.10
Age	177	19.87
Embarked	2	0.22

Pie Chart on missing values

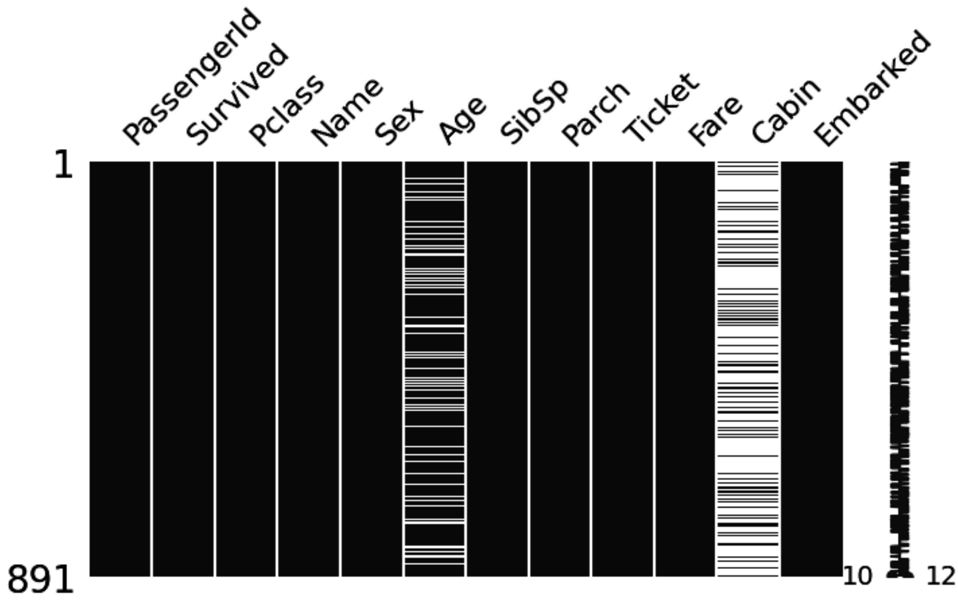
[20]:
```python
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0]
missing_values.sort_values(inplace=True)
missing_values.plot.pie(explode=[.1,.1,.1],
                        autopct='%1.1f%%',
                        shadow=True)
plt.title('Missing Values',font='Lucida Calligraphy')

```



Matrix on missing values

[21]:
```
msno.matrix(df,figsize=(8,4))
plt.show()
```



Working on missing values

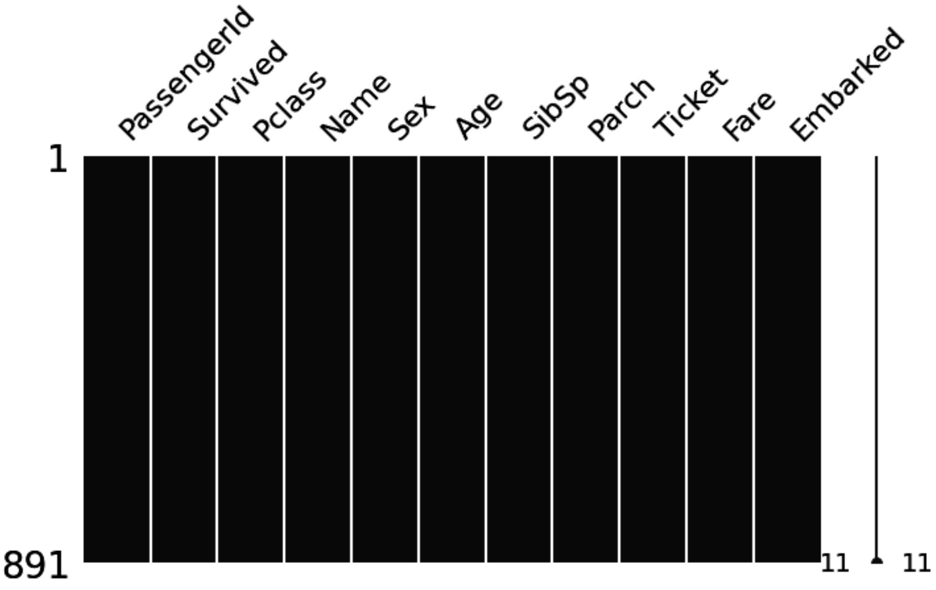
[24]:
```
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

[25]:
```
df['Embarked'] = df['Embarked'].fillna(method='bfill')
```

[26]:
```
df = df.drop(['Cabin'], axis=1)
```

[27]:

```
msno.matrix(df, figsize=(8,4))
plt.show()
```



[28]:

```
df.isnull().sum().sum()
```

[28…

```
0
```

Dropping unnecessary columns

[29]:

```
df = df.drop(['Name','Ticket'],axis=1)
```

[30]:

```
df.head()
```

[30…

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S

Transforming the categorical columns into numerical one

[31]:

```
df = pd.get_dummies(df, columns=['Sex','Embarked'], drop_first=True, dtype=int)
df.head()
```

[31…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	1	0	3	22.0	1	0	7.2500	1	0	1
1	2	1	1	38.0	1	0	71.2833	0	0	0
2	3	1	3	26.0	0	0	7.9250	0	0	1
3	4	1	1	35.0	1	0	53.1000	0	0	1
4	5	0	3	35.0	0	0	8.0500	1	0	1

[34]:

```
X = df.drop('Survived',axis=1)
y = df['Survived']
```

Splitting the data into Train and Test Set

[36]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=35)
```

Normalisation

[37]:

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

X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)
```

[38]:

```
display(X_train.head())
display(X_test.head())
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	0.068012	0.820268	0.655112	-0.473165	-0.470393	-0.507031	0.713074	-0.31696	0.601146
1	-0.953297	0.820268	0.340576	-0.473165	-0.470393	-0.518447	0.713074	-0.31696	0.601146
2	-0.506717	-0.386407	1.913254	-0.473165	-0.470393	-0.363866	0.713074	-0.31696	0.601146
3	0.588375	-1.593083	1.520085	0.377576	-0.470393	0.520444	0.713074	-0.31696	-1.663489
4	-0.036837	-1.593083	2.699594	0.377576	4.722159	4.765351	0.713074	-0.31696	0.601146

	PassengerId	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	0.203928	0.820268	-0.996201	-0.473165	-0.470393	-0.473814	0.713074	-0.31696	0.601146
1	-0.118386	-0.386407	-0.917567	-0.473165	2.125883	-0.384465	-1.402379	-0.31696	0.601146
2	-0.060137	-0.386407	0.969647	0.377576	-0.470393	-0.116675	-1.402379	-0.31696	0.601146
3	1.683468	0.820268	0.261942	-0.473165	-0.470393	-0.489608	0.713074	-0.31696	0.601146
4	-1.547443	0.820268	-0.917567	0.377576	-0.470393	-0.285589	-1.402379	-0.31696	0.601146

ML Process with Logistic Regression

[39]:

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

log_training = round(logreg.score(X_train, y_train)*100,2)
log_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Accuracy: ',log_training)
print('Model Accuracy Score: ',log_accuracy)
```

```
Training Accuracy:  80.06
Model Accuracy Score:  78.77
```

ML Process with Support Vector Machine

[40]:

```
svc=SVC()
svc.fit(X_train,y_train)
y_pred = svc.predict(X_test)

svc_training = round(svc.score(X_train,y_train)*100,2)
svc_accuracy = round(accuracy_score(y_pred, y_test)*100,2)

print('Training Accuracy: ', svc_training)
print('Model Accuracy Score: ', svc_accuracy)
```

```
Training Accuracy:  83.99
Model Accuracy Score:  83.8
```

ML Process with KNeighbors Classifier

[41]:

```
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)

knn_training = round(knn.score(X_train,y_train)*100,2)
knn_accuracy = round(accuracy_score(y_pred, y_test)*100,2)

print('Training Accuracy: ',knn_training)
print('Model Accuracy Score: ', knn_accuracy)
```

```
Training Accuracy:  85.96
Model Accuracy Score:  81.01
```

ML Process with Naive Bayes

[42]:

```
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
y_pred = gaussian.predict(X_test)

gaussin_training = round(gaussian.score(X_train, y_train)*100,2)
gaussin_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Accuracy: ', gaussin_training)
print('Model Accuracy Score: ',gaussin_accuracy)
```

```
Training Accuracy:  79.78
Model Accuracy Score:  74.86
```

ML Process with Linear SVM

[43]:

```
linear_svc = LinearSVC()
linear_svc.fit(X_train, y_train)
y_pred = linear_svc.predict(X_test)

linear_training = round(linear_svc.score(X_train,y_train)*100,2)
linear_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Accuracy: ', linear_training)
print('Model Accuracy Score: ',linear_accuracy)
```

```
Training Accuracy:  80.2
Model Accuracy Score:  79.33
```

ML Process with Stochastic Gradient Descent

[44]:

```
sgd = SGDClassifier()
sgd.fit(X_train,y_train)
y_pred = sgd.predict(X_test)

sgd_training = round(sgd.score(X_train,y_train)*100,2)
sgd_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Score: ',sgd_training)
print('Model Accuracy Score: ',sgd_accuracy)
```

```
Training Score:  75.7
Model Accuracy Score:  75.98
```

ML Process with Decision Tree Classifier

[46]:

```
decision = DecisionTreeClassifier()
decision.fit(X_train,y_train)
y_pred=decision.predict(X_test)

decision_training = round(decision.score(X_train,y_train)*100,2)
decision_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Accuracy: ', decision_training)
print('Model Accuracy Score: ', decision_accuracy)
```

```
Training Accuracy:  100.0
Model Accuracy Score:  73.74
```

ML Process with Random Forest Classifier

[47]:

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train,y_train)
y_pred = random_forest.predict(X_test)
random_forest.score(X_train,y_train)

random_forest_training = round(random_forest.score(X_train, y_train)*100,2)
random_forest_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Accuracy: ',random_forest_training)
print('Model Accuracy Score: ',random_forest_accuracy)
```

```
Training Accuracy:  100.0
Model Accuracy Score:  82.12
```

ML Process with XGBOOST

[48]:

```
xgb = Xgb.XGBClassifier()
xgb.fit(X_train,y_train)
y_pred = xgb.predict(X_test)
xgb.score(X_train,y_train)

xgb_training = round(xgb.score(X_train,y_train)*100,2)
xgb_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

print('Training Accuracy: ',xgb_training)
print('Model Accuracy Score: ',xgb_accuracy)
```

```
Training Accuracy:  100.0
Model Accuracy Score:  78.21
```

[49]:

```
models = pd.DataFrame({
    'Model':[
        'Logistic Regression','Support Vector Machine','KNN','GaussianNB','Linear SVC','Stochastic Gradient Decent','Decisi
    ],
    'Training Accuracy':[
        log_training,svc_training,knn_training,gaussin_training,linear_training,sgd_training,decision_training,random_fores
    ],
    'Model Accuracy Score':[
        log_accuracy, svc_accuracy, knn_accuracy, gaussin_accuracy, linear_accuracy,sgd_accuracy, decision_accuracy, random
        xgb_accuracy
    ]
})
```

Sorting the models by the accuracy

[50]:

```
models.sort_values(by='Training Accuracy',ascending=False)
```

[50…

	Model	Training Accuracy	Model Accuracy Score
6	Decision Tree	100.00	73.74
7	RandomForest	100.00	82.12
8	XGBClassifier	100.00	78.21
2	KNN	85.96	81.01
1	Support Vector Machine	83.99	83.80
4	Linear SVC	80.20	79.33
0	Logistic Regression	80.06	78.77
3	GaussianNB	79.78	74.86
5	Stochastic Gradient Decent	75.70	75.98

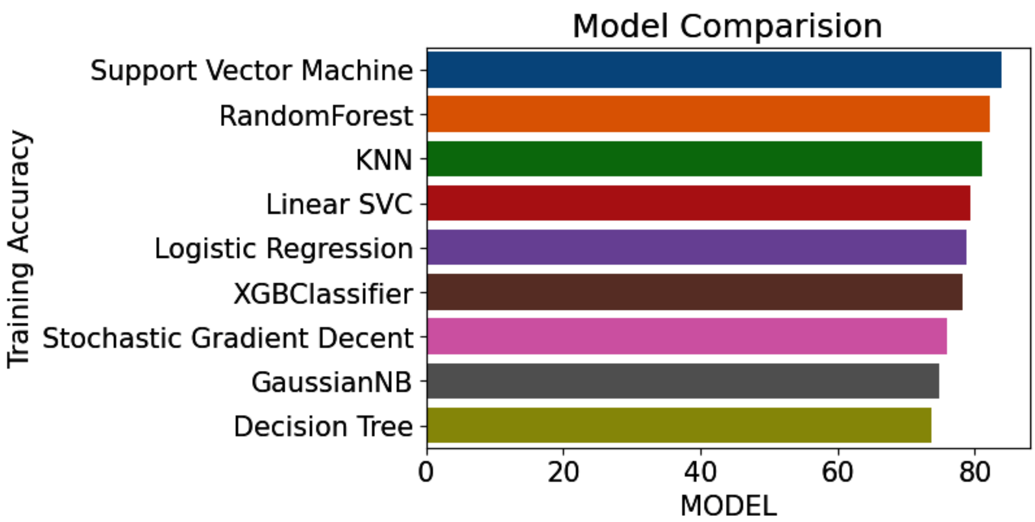
Visualisation on Model Comparison

[51]:

```
models = models.sort_values(by='Model Accuracy Score',ascending=False)[:20]
plt.figure(figsize=(6,4))
sns.barplot(y='Model',x='Model Accuracy Score',data=models)
plt.title('Model Comparision')
plt.xlabel('MODEL')
plt.ylabel('Training Accuracy')
```

[51…

```
Text(0, 0.5, 'Training Accuracy')
```



[208]:

```
y = df['Survived']

features = ['Pclass','SibSp','Parch']
X = pd.get_dummies(df[features])
X_test = pd.get_dummies(df[features])

model = SVC()
model.fit(X,y)
predictions = model.predict(X_test)

output = pd.DataFrame({'PassangerID': df.PassengerId, 'Survived': predictions})
output.to_csv('submission.csv',index=False)
print('Your submission was successfully saved !')
```

```
Your submission was successfully saved !
```

We successfully saved our ML Module.