TITANIC DATASET ANALYSIS

In this project I will be creating a machine learning model on the famous Titanic dataset, which is used by many people all over the world. It provides information on the fate of passengers on the Titanic, summarized according to economic status (class), sex, age and survival.

The “Titanic: Machine Learning from Disaster” Competition. In this challenge, we are asked to predict whether a passenger on the titanic would have been survived or not.

https://www.kaggle.com/c/titanic/data

IMPORTING THE LIBRARIES

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

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

UPLOADING THE DATA TO GOOGLE DRIVE

In [2]:
```
from google.colab import files
uploaded=files.upload()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving test.csv to test.csv
Saving train.csv to train.csv
```

LOADING THE DATA

In [3]:
```
import io
test=pd.read_csv(io.BytesIO(uploaded["test.csv"]))
train=pd.read_csv(io.BytesIO(uploaded["train.csv"]))
```

EDA

View the shape of the dataset

In [4]:
```
train.shape
```

Out[4]:
```
(891, 12)
```

In [5]:
```
test.shape
```

Out[5]:
```
(418, 11)
```

View the features of the dataset

In [6]:
```
train.columns
```

Out[6]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibS
p',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [7]:
```
test.columns
```

Out[7]:
```
Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

View the top few rows of the test and train dataset

In [8]:
```
train.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N

In [9]:
```
test.head()
```

Out[9]:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

View the statistical data of the datasets

In [10]:
```
train.describe()
```

Out[10]:

	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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [11]:
```
test.describe()
```

Out[11]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

View the types of variables

In [12]:
```
train.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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 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
```

In [13]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

How many survived

In [14]:
```
train['Survived'].value_counts()
```

Out[14]:
```
0    549
1    342
Name: Survived, dtype: int64
```

check for missing values

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [15]:
```
train.isnull().sum()
```

Out[15]:
```
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
```

In [16]:
```
test.isnull().sum()
```

Out[16]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

PLOTTING SOME GRAPHS

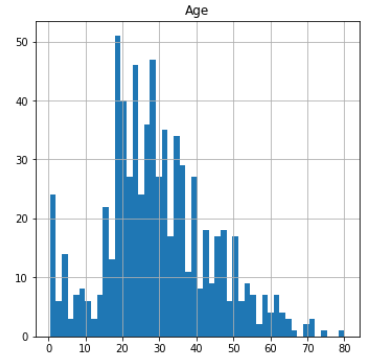
Plot relation between all the variables

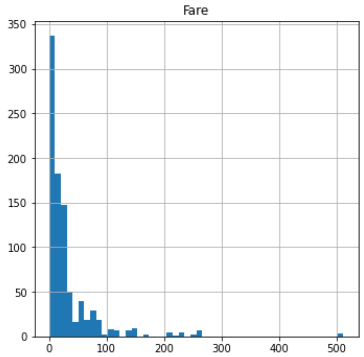
In [17]:
```
train.hist(bins=50,figsize=(20,20))
```

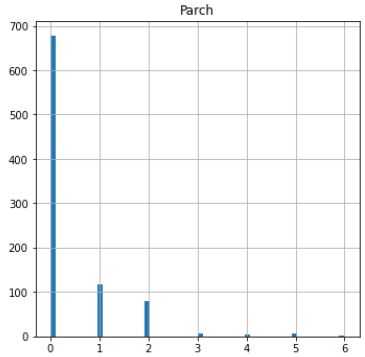
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

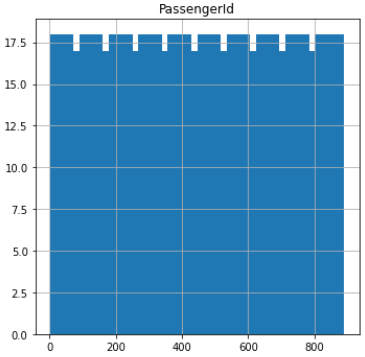
Out[17]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7feec17d9320
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7feec17ae5c0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7feec1765828
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7feec1718a90
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7feec174bcf8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7feec16fff60
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7feec16c0208
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7feec1674438
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7feec16744a8
>]],
      dtype=object)
```

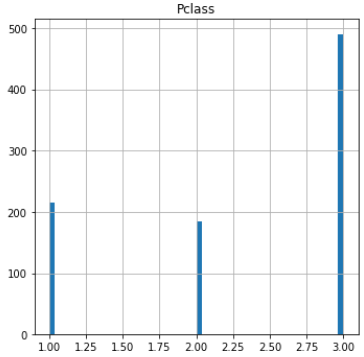
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

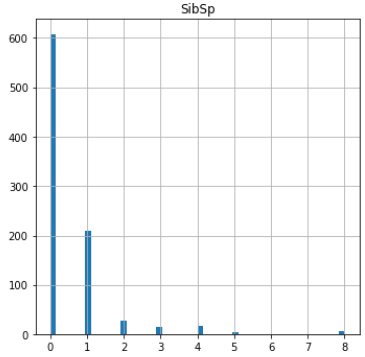


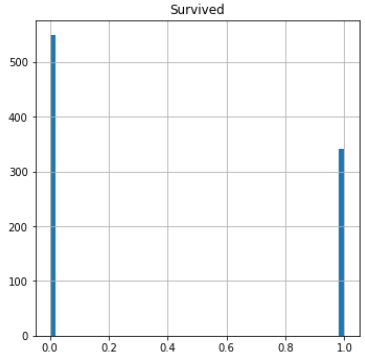








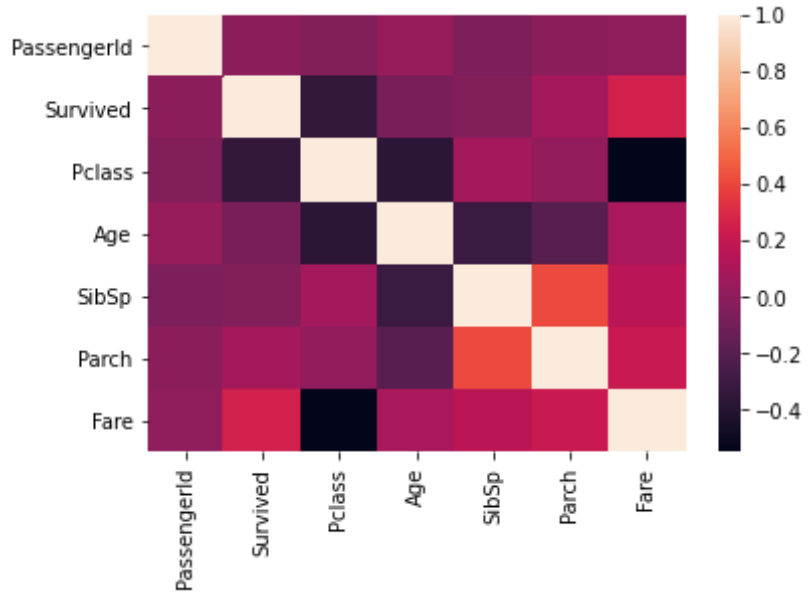




Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [18]:
```
train.corr()
sns.heatmap(train.corr())
```

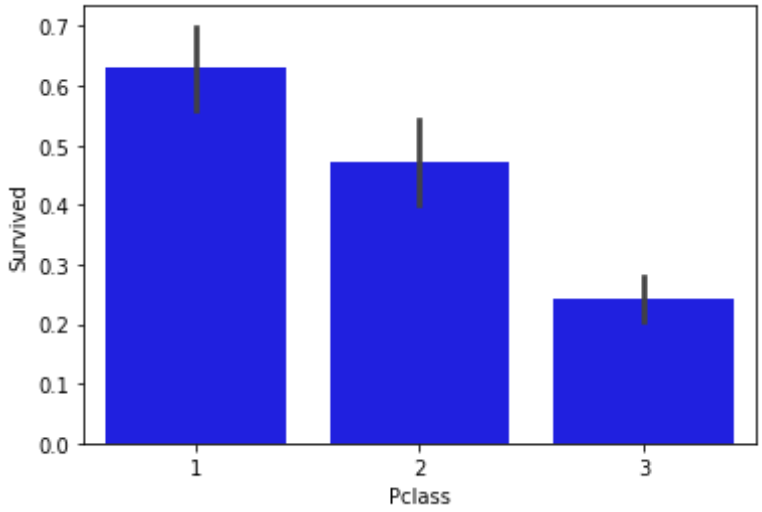
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7feec10d7908>



In [19]:
```
sns.barplot(x="Pclass", y="Survived", data=train, color="b")
```

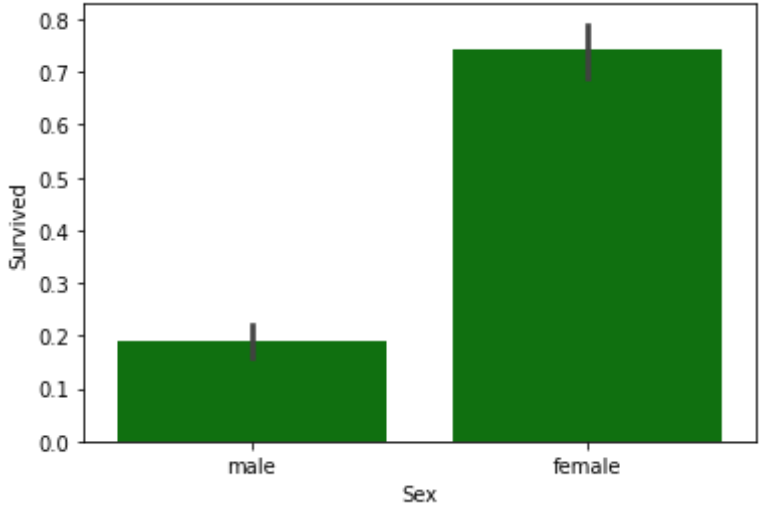
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7feebf97e3c8>

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [20]:
```
sns.barplot(x='Sex',y='Survived',data=train,color='g')
```

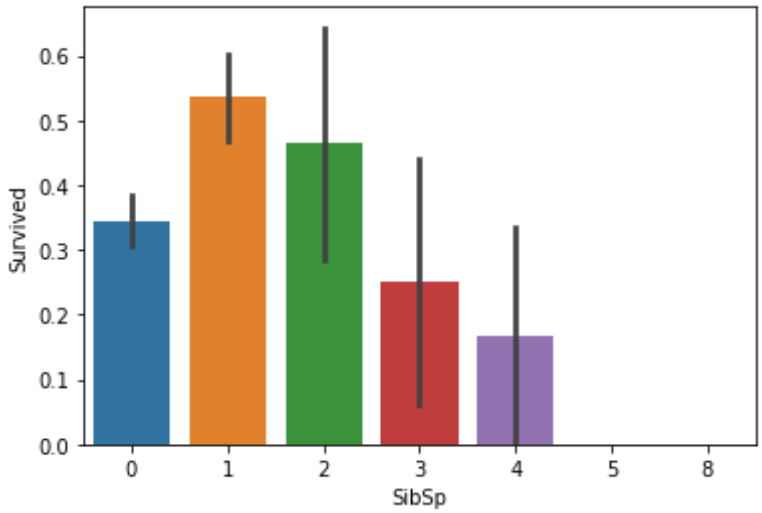
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7feebd09af28>



In [21]:
```
sns.barplot(x="SibSp", y="Survived", data=train)
```

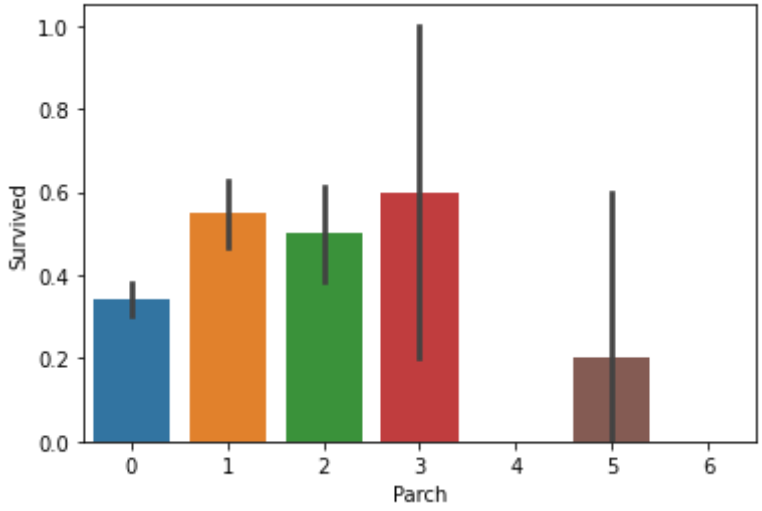
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7feebd010d30>

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In [22]:
```
sns.barplot(x="Parch", y="Survived", data=train)
```

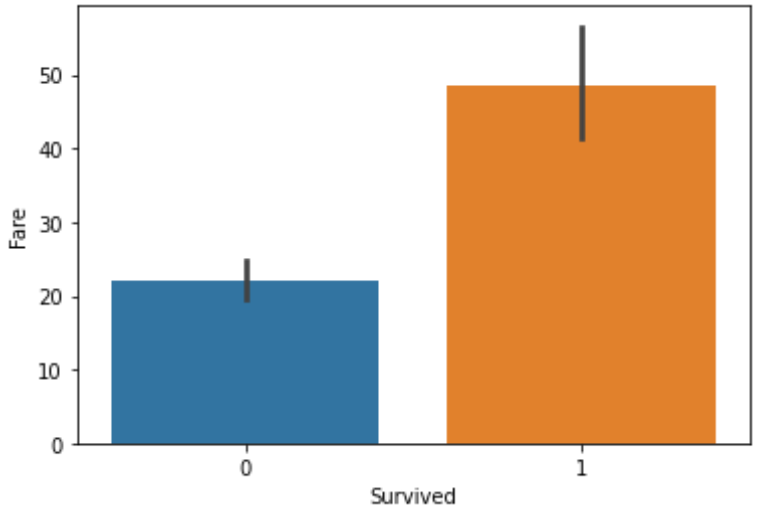
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7feebcfedda0>



In [23]:
```
sns.barplot(x="Survived", y="Fare", data=train)
```

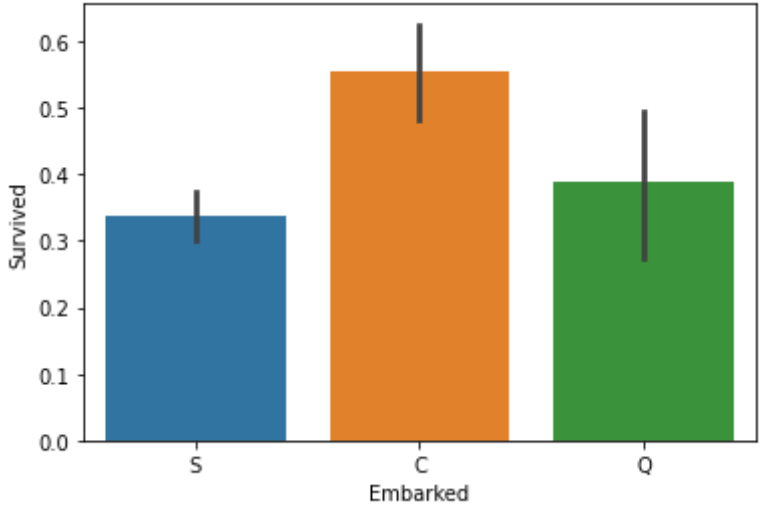
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7feebcf70f98>



In [24]:
```
sns.barplot(x="Embarked", y="Survived", data=train)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7feebcf279b0>



In [25]:
```
train=train.drop("PassengerId",axis=1)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
train=train.drop("Name",axis=1)
train=train.drop("Ticket",axis=1)
```

In [26]:
```
test_passenger_Id= test["PassengerId"]
test=test.drop("PassengerId",axis=1)
test=test.drop("Name",axis=1)
test=test.drop("Ticket",axis=1)
```

In [27]:
```
train=train.drop("Cabin",axis=1)
```

In [28]:
```
test=test.drop("Cabin",axis=1)
```

In [29]:
```
train.head()
```

Out[29]:

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

In [30]:
```
train = train.fillna(train['Age'].mean())
train= train.fillna(train['Embarked'].mode())

test= test.fillna(test['Age'].mean())
test = test.fillna(test['Fare'].median())
```

In [31]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
train['Sex'] = labelencoder.fit_transform(train['Sex'].astype(str))
train['Embarked'] = labelencoder.fit_transform(train['Embarked'].astype
(str))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
test['Sex'] = labelencoder.fit_transform(test['Sex'].astype(str))
test['Embarked'] = labelencoder.fit_transform(test['Embarked'].astype(s
tr))

test.head()
```

Out[31]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	34.5	0	0	7.8292	1
1	3	0	47.0	1	0	7.0000	2
2	2	1	62.0	0	0	9.6875	1
3	3	1	27.0	0	0	8.6625	2
4	3	0	22.0	1	1	12.2875	2

In [32]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier

X_train,X_val,Y_train,Y_val = train_test_split(train.drop('Survived',ax
is=1),train["Survived"],test_size=0.2, random_state=42)
```

In [33]:
```
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [34]:
```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)

Y_prediction = random_forest.predict(X_val)

random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100,
2)
acc_random_forest
```

Out[34]:
```
98.03
```

In [35]:
```
model = XGBClassifier()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_val)
print(accuracy_score(Y_pred,Y_val))
```

```
0.8156424581005587
```

In [36]:
```
model = LinearSVC(random_state=0, tol=1e-5)
model.fit(X_train, Y_train)
Y_pred = model.predict(X_val)
print(accuracy_score(Y_pred,Y_val))
```

```
0.8156424581005587
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: Conver
genceWarning: Liblinear failed to converge, increase the number of iter
ations.
  "the number of iterations.", ConvergenceWarning)
```

In [37]:
```
sgd = linear_model.SGDClassifier(max_iter=5, tol=None)
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_val)

sgd.score(X_train, Y_train)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
acc_sgd
```

Out[37]: 71.07

In [38]:
```
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)

Y_pred = logreg.predict(X_val)

acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

Out[38]: 80.2

In [39]:
```
# KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_val)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

Out[39]: 83.43

In [40]:
```
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_val)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

Out[40]: 79.92

In [41]:
```
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)

Y_pred = linear_svc.predict(X_val)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
acc_linear_svc
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: Conver
genceWarning: Liblinear failed to converge, increase the number of iter
ations.
  "the number of iterations.", ConvergenceWarning)
```

Out[41]: 74.02

In [42]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_val)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100,
2)
acc_decision_tree
```

Out[42]: 98.03

In [43]:
```
results = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes',
              'Stochastic Gradient Decent',
              'Decision Tree'],
    'Score': [acc_linear_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian,
              acc_sgd, acc_decision_tree]})
result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Out[43]:

Score	Model
98.03	Random Forest
98.03	Decision Tree
83.43	KNN

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Score	Model
80.20	Logistic Regression
79.92	Naive Bayes
74.02	Support Vector Machines
71.07	Stochastic Gradient Decent

In [44]:
```
from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier(n_estimators=100)
scores = cross_val_score(rf, X_train, Y_train, cv=10, scoring = "accura
cy")
print("Scores:", scores)
print("Mean:", scores.mean())
print("Standard Deviation:", scores.std())
```

```
Scores: [0.81944444 0.76388889 0.76056338 0.83098592 0.8028169  0.77464
789
 0.76056338 0.78873239 0.77464789 0.88732394]
Mean: 0.7963615023474179
Standard Deviation: 0.03829641923669304
```

In [45]:
```
importances = pd.DataFrame({'feature':X_train.columns,'importance':np.r
ound(random_forest.feature_importances_,3)})
importances = importances.sort_values('importance',ascending=False).set
_index('feature')
importances.head(15)
```

Out[45]:

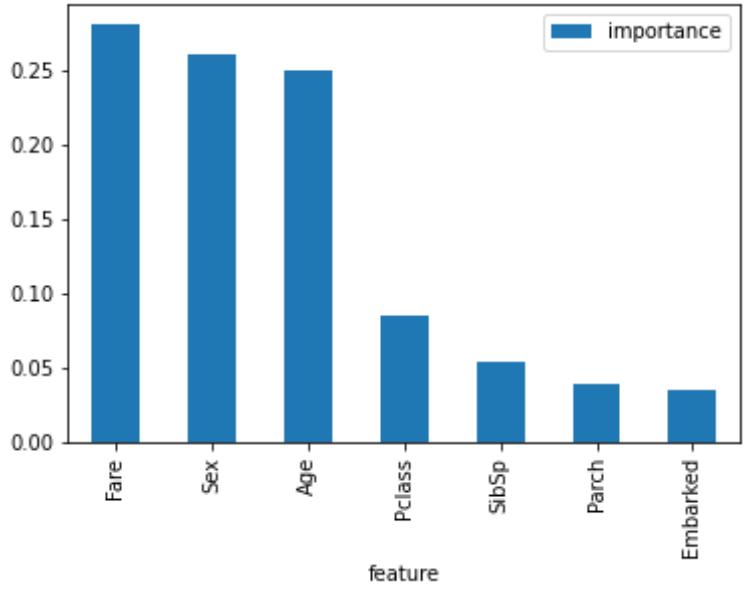
feature	importance
Fare	0.280
Sex	0.261
Age	0.249
Pclass	0.084

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

feature	importance
SibSp	0.053
Parch	0.039
Embarked	0.034

In [46]:
```
importances.plot.bar()
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7feeaf641b00>



In [ ]:
```

```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD