Machine Learning Laboratory

(410302)

BE Sem I Honors in AI/ML

Academic Year: 2021-22

Lab Assignment No. 7

Name: Aboli Marathe
Roll Number: 41301
Branch: Department of Computer Engineering

Problem Statement:

Write a program to solve a problem using Decision tree or Random forest algorithm

Lab Exercise 1

Use famous iris flower dataset from sklearn.datasets to predict flower species using random forest classifier.

1. Measure prediction score using default n_estimators (10)
2. Now fine tune your model by changing number of trees in your classifier and tell me what best score you can get using how many trees

```
from sklearn import datasets

iris = datasets.load_iris()

print(iris.target_names)

     ['setosa' 'versicolor' 'virginica']


print(iris.feature_names)

     ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

X, y = datasets.load_iris( return_X_y = True)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.70)
```

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
data = pd.DataFrame({'sepallength': iris.data[:, 0], 'sepalwidth': iris.data[:, 1],
                     'petallength': iris.data[:, 2], 'petalwidth': iris.data[:, 3],
                     'species': iris.target})
```

```
print(data.head())
```

```
   sepallength  sepalwidth  petallength  petalwidth  species
0          5.1         3.5          1.4         0.2        0
1          4.9         3.0          1.4         0.2        0
2          4.7         3.2          1.3         0.2        0
3          4.6         3.1          1.5         0.2        0
4          5.0         3.6          1.4         0.2        0
```

```
clf = RandomForestClassifier(n_estimators = 10)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
from sklearn import metrics
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred)*100,"%")
```

```
ACCURACY OF THE MODEL:  96.19047619047619 %
```

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred)*100, "%")
```

```
ACCURACY OF THE MODEL:  95.23809523809523 %
```

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 1000)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred)*100, "%")
```

```
ACCURACY OF THE MODEL:  96.19047619047619 %
```

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 5)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred)*100, "%")
```

```
ACCURACY OF THE MODEL:  97.14285714285714 %
```

Conclusion

Comparing the classifiers by their result, I found the best model with 5 trees giving an accuracy of 97.14285714285714 %.

Lab Exercise 2

Build decision tree model to predict survival of titanic based on certain parameters. .csv file is available to download at Kaggle or is also with lab assignment. In this file using following columns build a model to predict if person would survive or not,

1. Pclass 2. Sex 3. Age 4. Fare

Calculate score of your model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df = pd.read_csv('titanic.csv')
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th	female	38.0	1	0	PC 17599	71.

```
df.isnull()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	

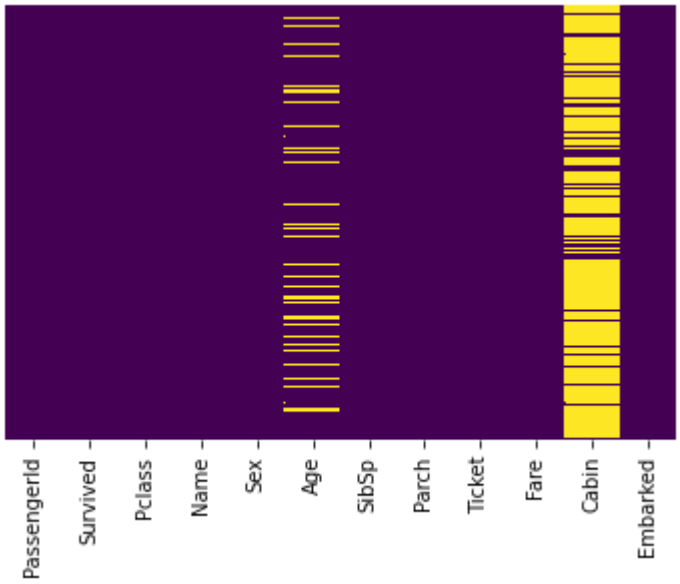
Exploratory Data Analysis

Missing Data

Use seaborn to create a simple heatmap to see where we are missing data!

```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

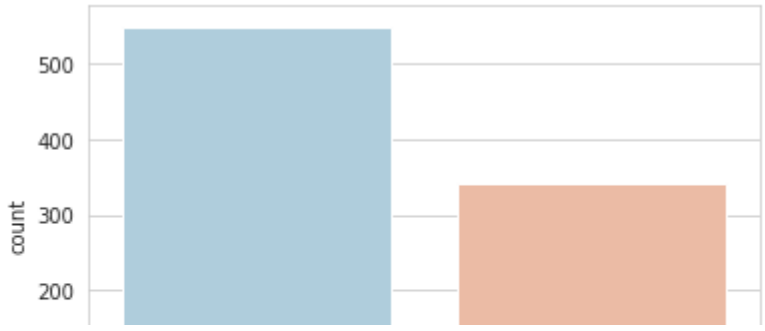
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3e436610>
```



Roughly 20 percent of the Age data is missing. The proportion of Age missing is likely small enough for reasonable replacement with some form of imputation. Looking at the Cabin column, it looks like we are just missing too much of that data to do something useful with at a basic level.

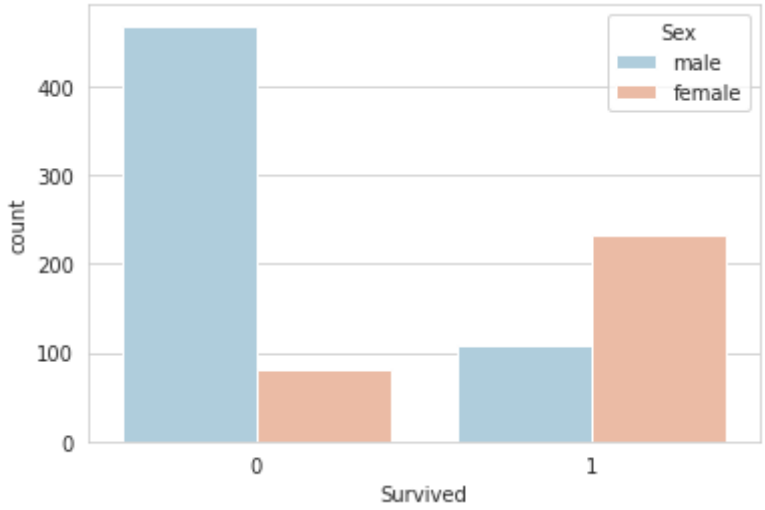
```
sns.set_style('whitegrid')
sns.countplot(x='Survived', data=df, palette='RdBu_r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3e0a1150>
```



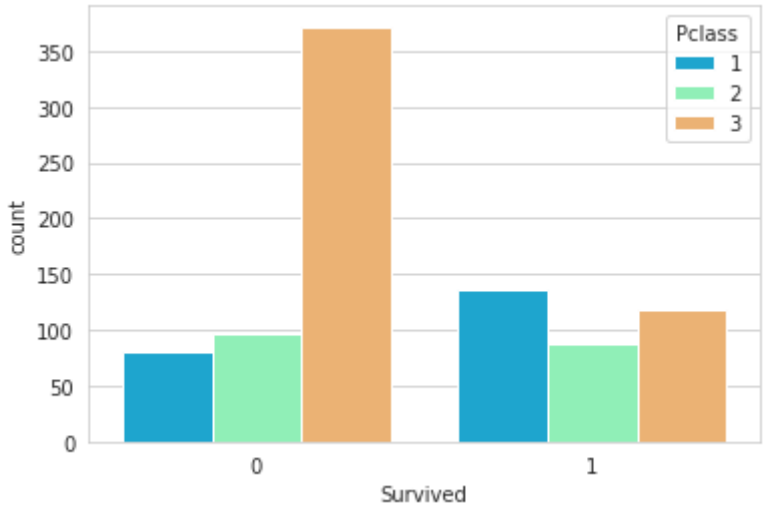

```
sns.set_style('whitegrid')
sns.countplot(x='Survived', hue='Sex', data=df, palette='RdBu_r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3e436390>
```



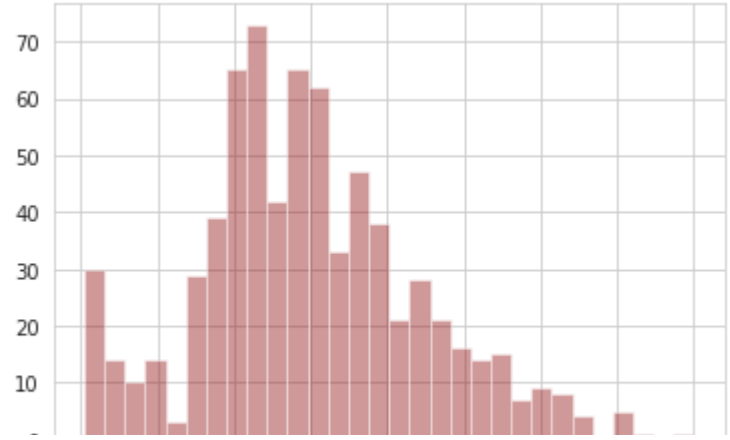

```
sns.set_style('whitegrid')
sns.countplot(x='Survived', hue='Pclass', data=df, palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3dc71990>
```



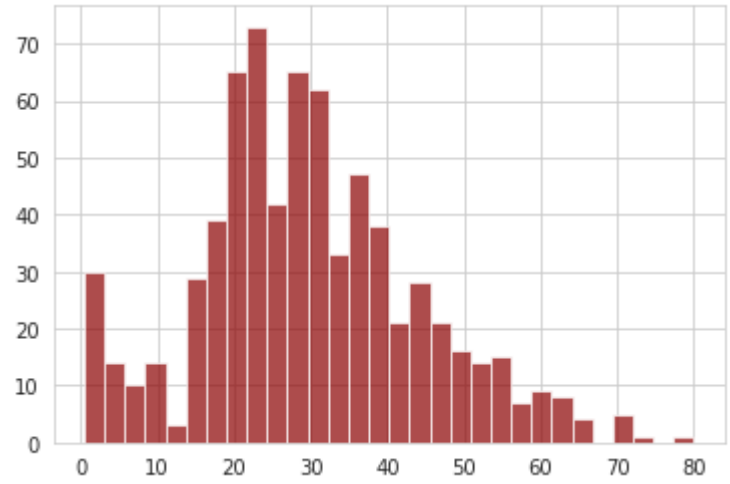

```
sns.distplot(df['Age'].dropna(), kde=False, color='darkred', bins=30)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3dff7150>
```



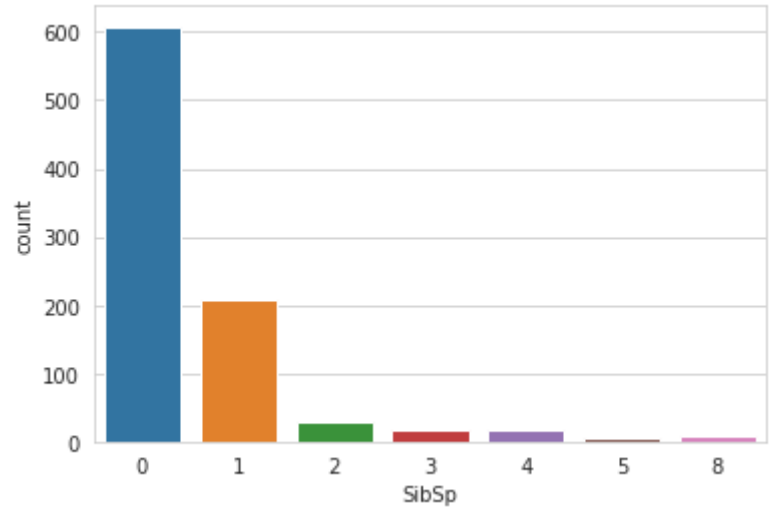

```
df['Age'].hist(bins=30, color='darkred', alpha=0.7)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3e28ded0>
```



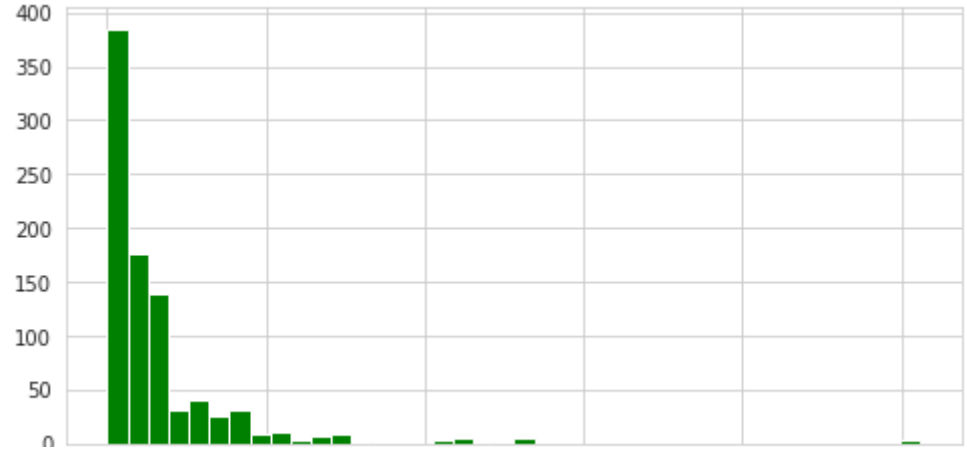

```
sns.countplot(x='SibSp', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3df5d3d0>
```




```
df['Fare'].hist(color='green', bins=40, figsize=(8,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3de8d050>
```

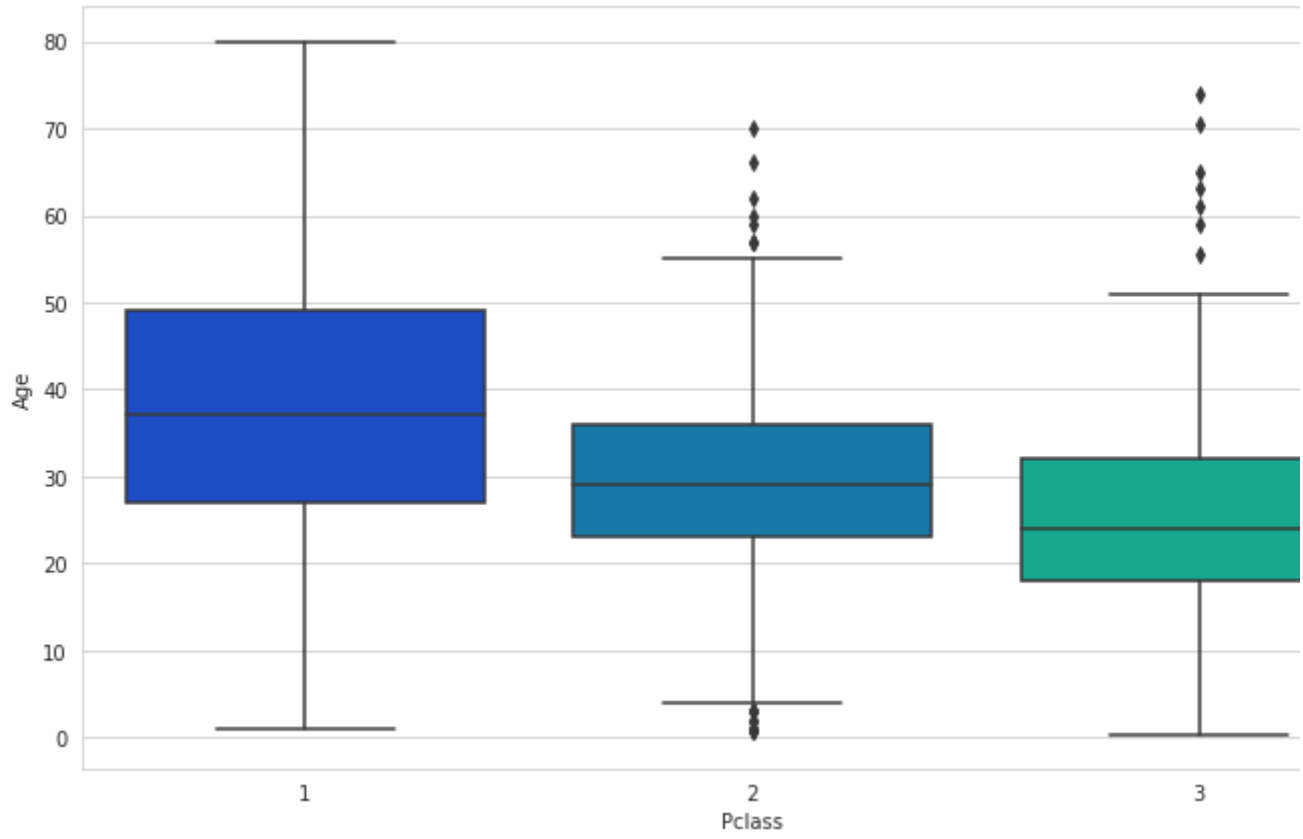


Data Cleaning

Fill in missing age data instead of dropping the missing age data rows. One way to do this is by filling in the mean age of all the passengers (imputation). However we can check the average age by passenger class. For example:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Pclass', y='Age', data=df, palette='winter')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3e0a1590>
```



Wealthier passengers in the higher classes tend to be older, which makes sense. We'll use these average age values to impute based on Pclass for Age.

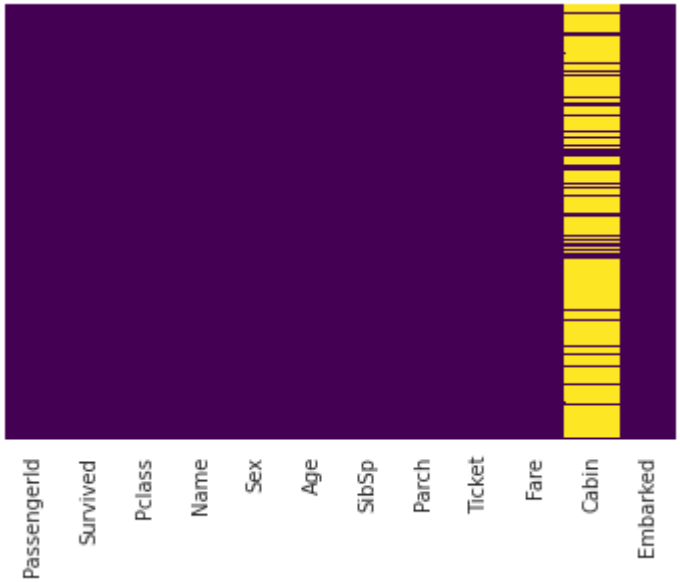
```
def impute_age(cols):
```

```
    Age = cols[0]
    Pclass = cols[1]
    if pd.isnull(Age):
        if Pclass == 1:
            return 37
        elif Pclass == 2:
            return 29
        else:
            return 24
    else:
        return Age
```

```
df['Age'] = df[['Age', 'Pclass']].apply(impute_age, axis=1)
```

```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2f3dd52490>
```



Drop the Cabin column and the row in Embarked that is NaN.

```
df.drop('Cabin', axis=1, inplace=True)
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
				Heikkinen					STON/O2

```
df.dropna(inplace=True)
```

Converting Categorical Features

Convert categorical features to dummy variables using pandas.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  889 non-null    int64  
 1   Survived     889 non-null    int64  
 2   Pclass       889 non-null    int64  
 3   Name         889 non-null    object 
 4   Sex          889 non-null    object 
 5   Age          889 non-null    float64
 6   SibSp        889 non-null    int64  
 7   Parch        889 non-null    int64  
 8   Ticket       889 non-null    object 
 9   Fare         889 non-null    float64
 10  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 83.3+ KB
```

```
sex = pd.get_dummies(df['Sex'], drop_first=True)
embark = pd.get_dummies(df['Embarked'], drop_first=True)
```

```
df.drop(['Sex', 'Embarked', 'Name', 'Ticket'], axis=1, inplace=True)
```

```
df = pd.concat([df, sex, embark], axis=1)
```

```
df.head()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	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

```
df.drop('PassengerId', axis=1, inplace=True)
```

```
df.head()
```

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

```
y = df['Survived']
```

```
X = df.iloc[:,1:]
```

```
X.head()
```

	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	3	22.0	1	0	7.2500	1	0	1
1	1	38.0	1	0	71.2833	0	0	0
2	3	26.0	0	0	7.9250	0	0	1
3	1	35.0	1	0	53.1000	0	0	1
4	3	35.0	0	0	8.0500	1	0	1

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_pred)
```

```
array([[128,  29],
       [ 33,  77]])
```

```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_pred)
```

```
0.7677902621722846
```

Conclusion

The accuracy of predicting the chances of survival on the Titanic based on these parameters is 76.77%.