titanic (2)

July 2, 2024

1 Titanic Data Analysis and Classification

1.0.1 Imports and Data Preparation

[1]:
```
%%capture
!pip install tensorflow;
```

[2]:
```
import numpy as np
import pandas as pd
from sklearn.feature_selection import chi2
from sklearn.feature_selection import f_classif
from sklearn import tree
import re
import math
from tensorflow import keras
from sklearn.preprocessing import MinMaxScaler
```

[3]:
```
train = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')
```

[4]:
```
data = pd.concat([train, test])
```

[5]:
```
data[7:10]
```

[5]:
```
   PassengerId  Survived  Pclass  \
7            8       0.0       3
8            9       1.0       3
9           10       1.0       2

                                                Name     Sex   Age  SibSp  \
7                     Palsson, Master. Gosta Leonard    male   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1

   Parch  Ticket     Fare Cabin Embarked
7      1  349909  21.0750   NaN        S
8      2  347742  11.1333   NaN        S
```

```
9       0  237736  30.0708   NaN        C
```

[6]:
```
data['InTrain'] = data['PassengerId'].apply(lambda x: True if x <= len(train)␣
 ↪else False)
```

1.0.2 Ticket Class

Summary

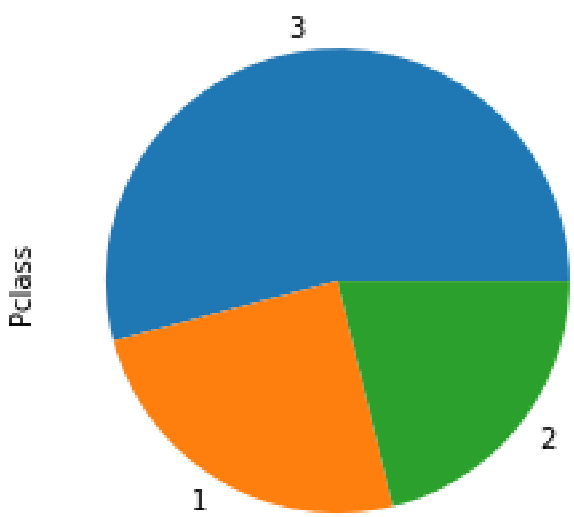
[7]:
```
data['Pclass'].describe()
```

[7]:
```
count    1309.000000
mean        2.294882
std         0.837836
min         1.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         3.000000
Name: Pclass, dtype: float64
```

Visualization

[8]:
```
data['Pclass'].value_counts().plot.pie()
```

[8]: <AxesSubplot:ylabel='Pclass'>



Correlation

```
[9]: corr = chi2(pd.DataFrame(data['Pclass'][0:len(train)]), data['Survived'][0:
     ↪len(train)])
     print(f'Chi2 = {corr[0]}\nP-value = {corr[1]}')
```

```
Chi2 = [30.87369944]
P-value = [2.75378563e-08]
```

1.0.3 Sex

Summary

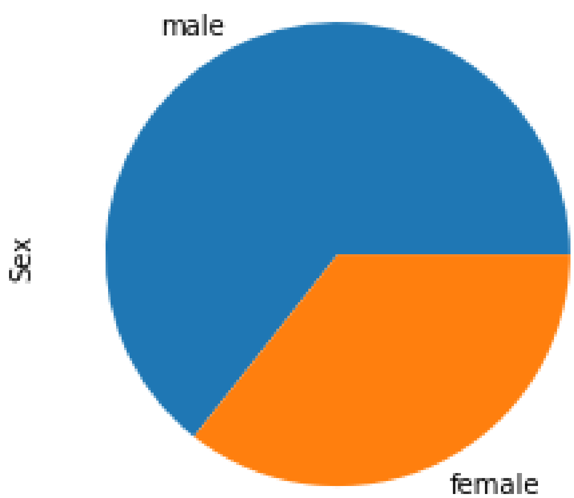
```
[10]: data['Sex'].describe()
```

```
[10]: count      1309
      unique        2
      top        male
      freq        843
      Name: Sex, dtype: object
```

Visualization

```
[11]: data['Sex'].value_counts().plot.pie()
```

```
[11]: <AxesSubplot:ylabel='Sex'>
```



Correlation

```
[12]: data['Sex'] = data['Sex'].apply(lambda x: 1 if x == 'male' else 0)
```

```
corr = chi2(pd.DataFrame(data['Sex'][0:len(train)]), data['Survived'][0:
 ↪len(train)])
print(f'Chi2 = {corr[0]}\nP-value = {corr[1]}')
```

```
Chi2 = [92.70244698]
P-value = [6.07783826e-22]
```

1.0.4 Fare

Summary

[13]:
```
data['Fare'].describe()
```

[13]:
```
count    1308.000000
mean       33.295479
std        51.758668
min         0.000000
25%         7.895800
50%        14.454200
75%        31.275000
max       512.329200
Name: Fare, dtype: float64
```

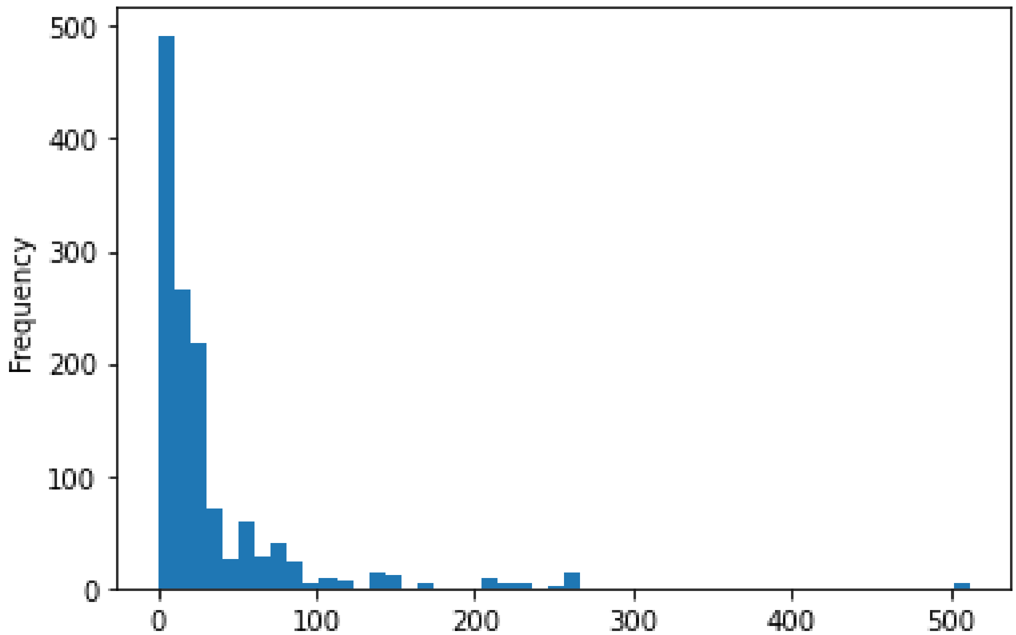
Imputation

[14]:
```
fare_avg = data['Fare'].mean()
data['Fare'].fillna(fare_avg, inplace = True);
```

Visualization

[15]:
```
data['Fare'].plot.hist(bins = 50)
```

[15]:
```
<AxesSubplot:ylabel='Frequency'>
```



Correlation

[16]:
```
corr = f_classif(pd.DataFrame(data['Fare'][0:len(train)]), data['Survived'][0:
 ↪len(train)])
print(f'ANOVA = {corr[0]}\nP-value = {corr[1]}')
```

```
ANOVA = [63.03076423]
P-value = [6.12018934e-15]
```

1.0.5 Age / Age Group

Feature Engineering

[17]:
```
def assignAgeGroup(x):
    if x < 18:
        return 0
    elif x < 65:
        return 1
    elif x >= 65:
        return 2
    return 3

data['Age Group'] = data['Age'].apply(assignAgeGroup)
```

Summary

[18]:
```
data['Age'].describe()
```

[18]:
```
count    1046.000000
mean       29.881138
std        14.413493
min         0.170000
25%        21.000000
50%        28.000000
75%        39.000000
max        80.000000
Name: Age, dtype: float64
```

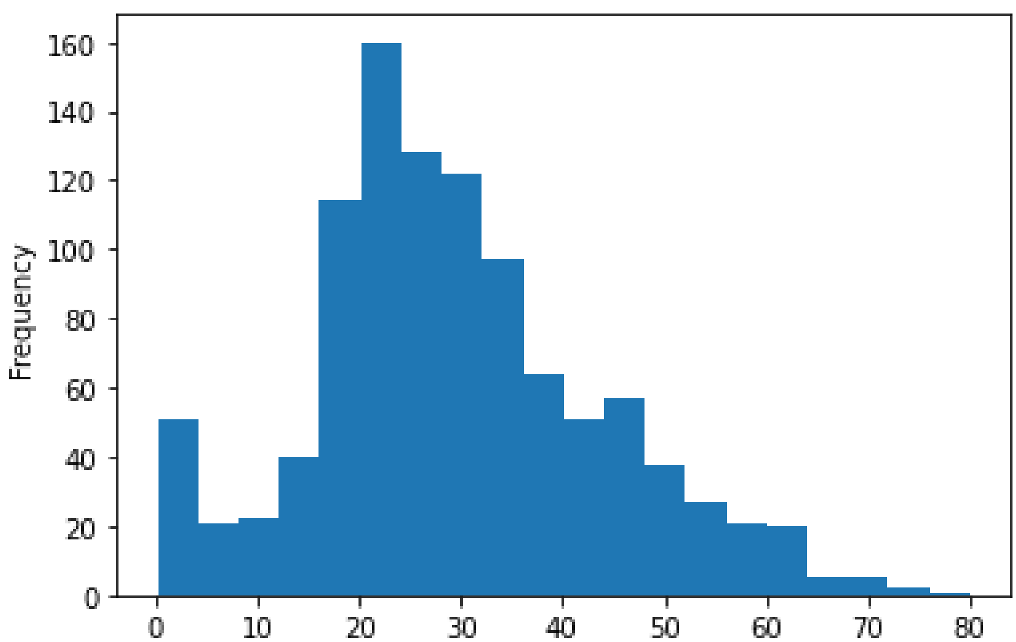
[19]:
```
data['Age Group'].describe()
```

[19]:
```
count    1309.000000
mean        1.294118
std         0.919449
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         3.000000
Name: Age Group, dtype: float64
```

Visualization

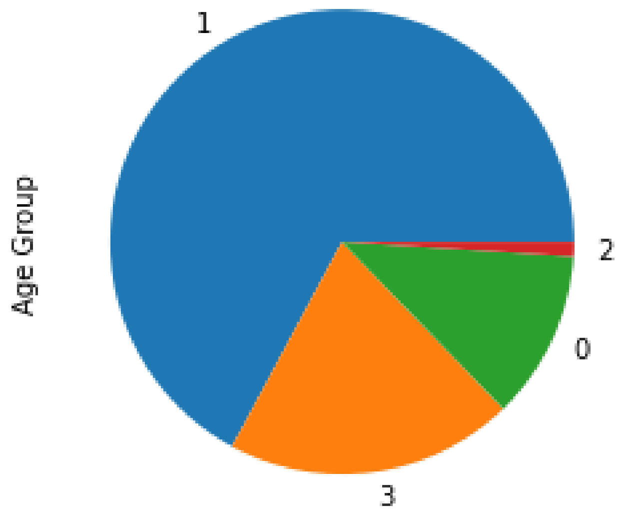
[20]:
```
data['Age'].plot.hist(bins = 20)
```

[20]:
```
<AxesSubplot:ylabel='Frequency'>
```



[21]:
```
data['Age Group'].value_counts().plot.pie()
```

[21]: <AxesSubplot:ylabel='Age Group'>



Correlation

[22]:
```
corr = f_classif(pd.DataFrame(data['Age'][0:len(train)]).dropna(),␣
↪data['Survived'][0:len(train)][data['Age'][0:len(train)].notna()])
print(f'Age:\nANOVA = {corr[0]}\nP-value = {corr[1]}')
```

```
Age:
ANOVA = [4.27119493]
P-value = [0.03912465]
```

[23]:
```
corr = chi2(pd.DataFrame(data['Age Group'][0:len(train)]), data['Survived'][0:
↪len(train)])
print(f'Age Group:\nChi2 = {corr[0]}\nP-value = {corr[1]}')
```

```
Age Group:
Chi2 = [10.28445646]
P-value = [0.00134156]
```

1.0.6 Cabin

Summary

[24]:
```
data['Cabin'].describe()
```

[24]:
```
count                295
unique               186
top          C23 C25 C27
freq                   6
Name: Cabin, dtype: object
```

Imputation

[25]:
```
data['Last Name'] = data['Name'].apply(lambda x: x.split()[0][0:-1])
```

[26]:
```
lastnames = data['Last Name'].unique()
cabins = ['U'] * len(data)

for lastname in lastnames:
    family = data[data['Last Name'] == lastname]
    if len(family['Cabin'].dropna()) != 0:
        cabin = family['Cabin'].dropna().tolist()[0]
        pIds = family['PassengerId'].tolist()
        for pId in pIds:
            cabins[pId-1] = cabin

data['Cabin'] = cabins
```

[27]:
```
data['Cabin'].describe()
```

[27]:
```
count      1309
unique      172
top           U
freq        977
Name: Cabin, dtype: object
```

Feature Engineering and Imputation

[28]:
```
data['Floor'] = data['Cabin'].apply(lambda x: x[0])
```

[29]:
```
room_sum = 0
for cabin in cabins:
    if len(cabin) != 1:
        room_sum += float(re.findall(r'\d+', cabin[-2:])[0])
room_avg = room_sum / 332

data['Room'] = data['Cabin'].apply(lambda x: room_avg if len(x) == 1 else␣
 ↪float(re.findall(r'\d+', x[-2:])[0]))
```

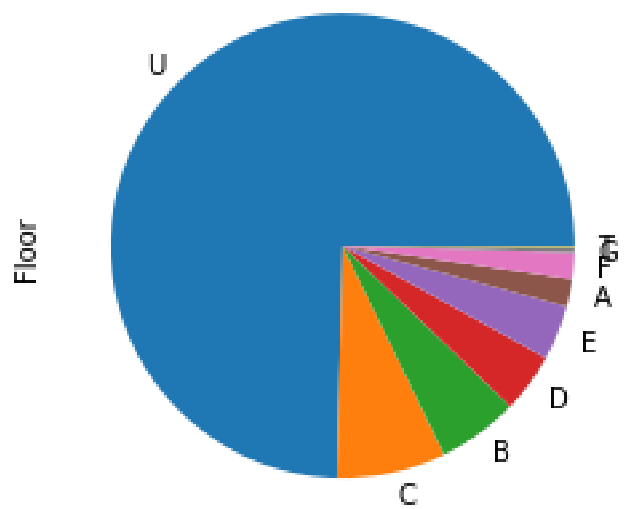
[30]:
```
data['Room'].describe()
```

[30]:
```
count    1309.000000
mean       37.311810
std        13.499729
min         1.000000
25%        37.141566
50%        37.141566
75%        37.141566
max        99.000000
Name: Room, dtype: float64
```

Visualization

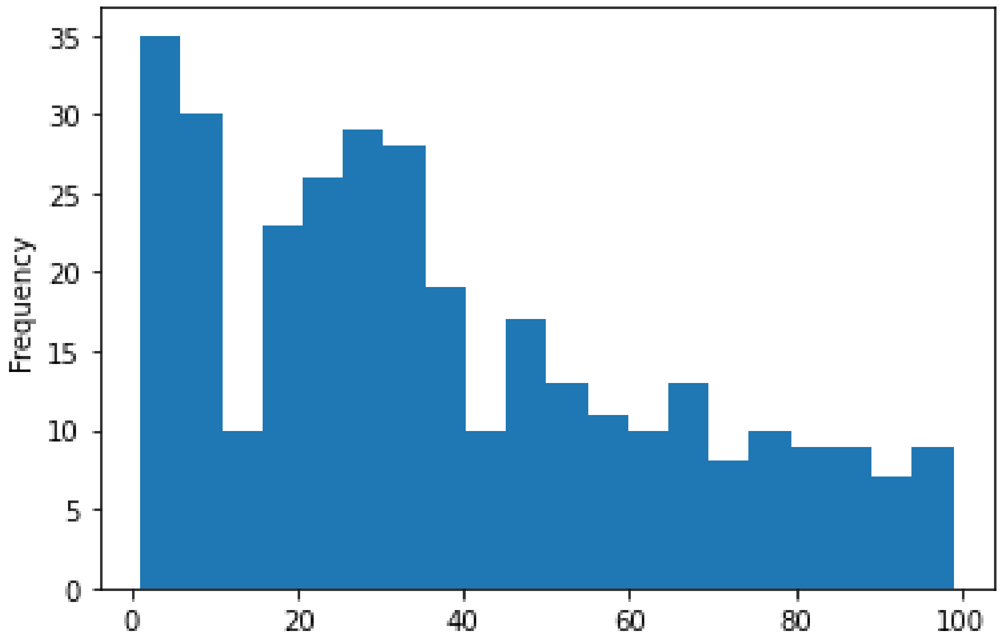
[31]:
```
data['Floor'][data['Floor'] != 3.5].value_counts().plot.pie()
```

[31]:
```
<AxesSubplot:ylabel='Floor'>
```



[32]:
```
data['Room'][data['Room'] != room_avg].plot.hist(bins = 20)
```

[32]:
```
<AxesSubplot:ylabel='Frequency'>
```



Correlation

```
[33]: data['Floor'] = data['Floor'].replace(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'T',␣
      ↪'U'], [0, 1, 2, 3, 4, 5, 6, 7, 3.5]);
      corr = f_classif(pd.DataFrame(data['Floor'][0:len(train)]), data['Survived'][0:
      ↪len(train)])
      print(f'Floor:\nANOVA = {corr[0]}\nP-value = {corr[1]}')
```

```
Floor:
ANOVA = [22.1482981]
P-value = [2.92578455e-06]
```

```
[34]: corr = f_classif(pd.DataFrame(data['Room'][0:len(train)][data['Room'][0:
      ↪len(train)] != room_avg]), data['Survived'][0:len(train)][data['Room'][0:
      ↪len(train)] != room_avg])
      print(f'Room:\nANOVA = {corr[0]}\nP-value = {corr[1]}')
```

```
Room:
ANOVA = [0.50850824]
P-value = [0.47652756]
```

1.0.7 Family Size

Feature Engineering

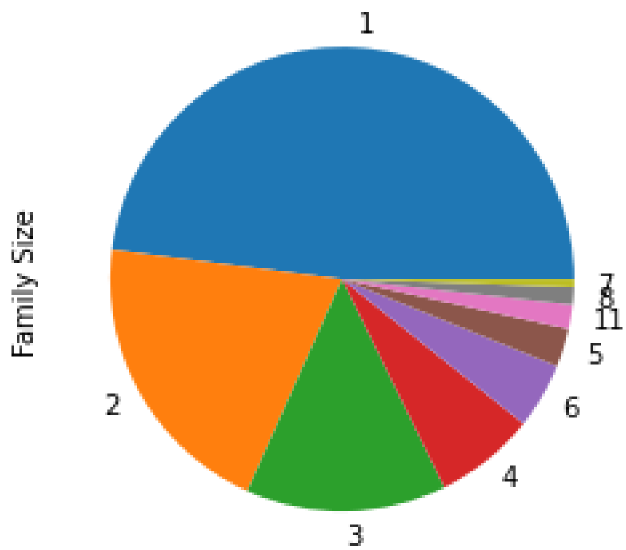
```
[35]: family_size = [None] * len(data)
```

```
for lastname in lastnames:
    family = data[data['Last Name'] == lastname]
    pIds = family['PassengerId'].tolist()
    for pId in pIds:
        family_size[pId-1] = len(family)

data['Family Size'] = family_size
```

Visualization

[36]:
```
data['Family Size'].value_counts().plot.pie()
```

[36]: <AxesSubplot:ylabel='Family Size'>



Correlation

[37]:
```
corr = f_classif(pd.DataFrame(data['Family Size'][0:len(train)]),␣
  ↪data['Survived'][0:len(train)])
print(f'Family Size:\nANOVA = {corr[0]}\nP-value = {corr[1]}')
```

```
Family Size:
ANOVA = [1.06329472]
P-value = [0.30274538]
```

1.0.8 Family Survived

Feature Engineering

[38]:

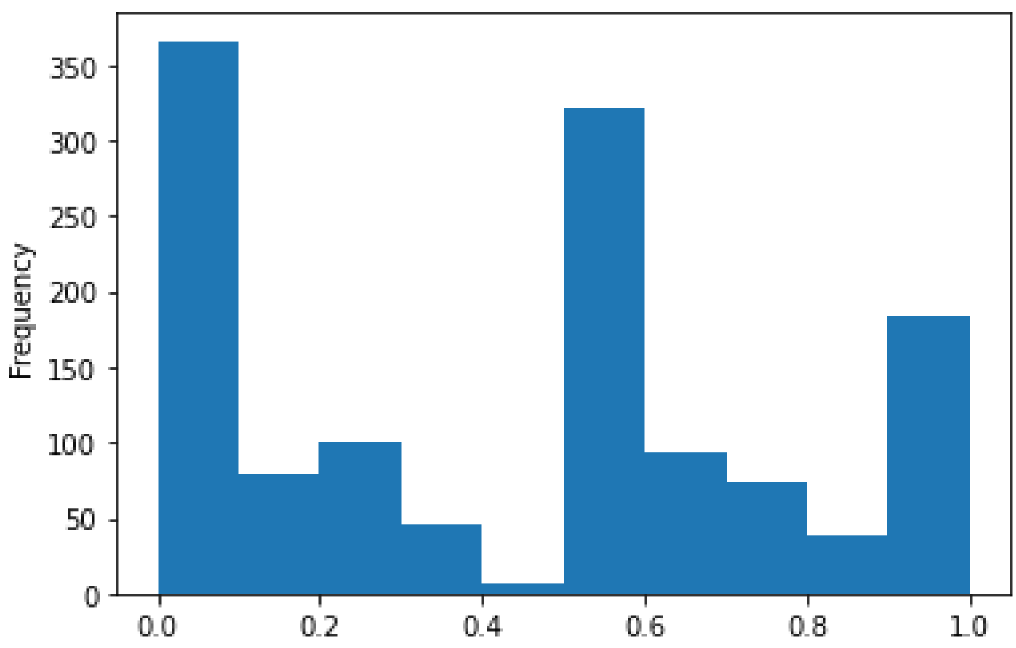
```
family_survived = [None] * len(data)

for lastname in lastnames:
    family = data[data['Last Name'] == lastname]
    survived = family['Survived'].tolist()
    avg_survived = 0
    for member in survived:
        if math.isnan(member):
            avg_survived += 0.5
        else:
            avg_survived += member
    avg_survived /= len(family)
    pIds = family['PassengerId'].tolist()
    for pId in pIds:
        family_survived[pId-1] = avg_survived

data['Family Survived'] = family_survived
```

Visualization

[39]:

```
data['Family Survived'].plot.hist(bins = 10)
```

[39]: <AxesSubplot:ylabel='Frequency'>



Correlation

[40]:
```
corr = f_classif(pd.DataFrame(data['Family Survived'][0:len(train)]),␣
↪data['Survived'][0:len(train)])
print(f'Family Survived:\nANOVA = {corr[0]}\nP-value = {corr[1]}')
```

```
Family Survived:
ANOVA = [3197.30644761]
P-value = [1.07334179e-296]
```

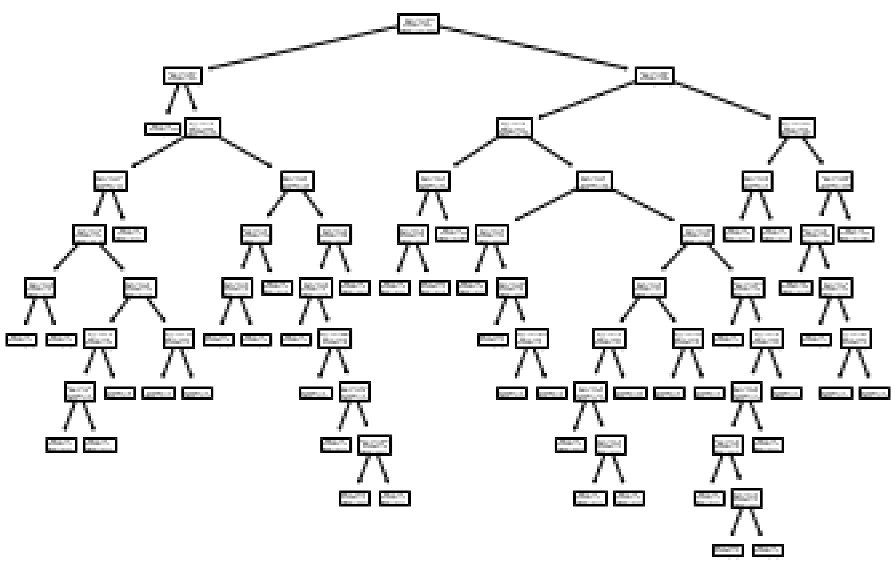
1.0.9 Model and Classification

Decision Tree

[41]:
```
final_train = data[['Pclass', 'Sex', 'Age Group', 'Floor', 'Family Survived']][:
↪len(train)]
final_test = data[['Pclass', 'Sex', 'Age Group', 'Floor', 'Family␣
↪Survived']][len(train):]
```

[42]:
```
d_tree = tree.DecisionTreeClassifier()
d_tree.fit(final_train, data['Survived'][:len(final_train)]);
```

[43]:
```
tree.plot_tree(d_tree);
```



[44]:
```
classifications = d_tree.predict(final_test)
final = pd.DataFrame()
final['Survived'] = classifications
final['PassengerId'] = data['PassengerId'][len(train):]
final.to_csv('submission.csv');
```

Neural Network

[45]:
```
# Standardize dataset
scaler = MinMaxScaler()
final_train = scaler.fit_transform(final_train)
final_test = scaler.transform(final_test)
```

[46]:
```
model = keras.Sequential()
model.add(keras.layers.Dense(128, activation = 'sigmoid'))
model.add(keras.layers.Dense(128, activation = 'sigmoid'))
model.add(keras.layers.Dense(1, activation = 'sigmoid'))
```

[47]:
```
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics =␣
↪['accuracy'])
```

[52]:
```
model.fit(x = final_train, y = data['Survived'][:len(final_train)], epochs = 15)
```

```
Epoch 1/15
28/28 [==============================] - 0s 963us/step - loss: 0.2088 -
accuracy: 0.9371
Epoch 2/15
28/28 [==============================] - 0s 889us/step - loss: 0.1904 -
accuracy: 0.9394
Epoch 3/15
28/28 [==============================] - 0s 778us/step - loss: 0.1730 -
accuracy: 0.9484
Epoch 4/15
28/28 [==============================] - 0s 815us/step - loss: 0.1628 -
accuracy: 0.9450
Epoch 5/15
28/28 [==============================] - 0s 778us/step - loss: 0.1493 -
accuracy: 0.9585
Epoch 6/15
28/28 [==============================] - 0s 852us/step - loss: 0.1422 -
accuracy: 0.9562
Epoch 7/15
28/28 [==============================] - 0s 778us/step - loss: 0.1348 -
accuracy: 0.9562
Epoch 8/15
28/28 [==============================] - 0s 778us/step - loss: 0.1340 -
accuracy: 0.9607
Epoch 9/15
28/28 [==============================] - 0s 815us/step - loss: 0.1227 -
accuracy: 0.9630
Epoch 10/15
28/28 [==============================] - 0s 778us/step - loss: 0.1218 -
accuracy: 0.9574
Epoch 11/15
```

```
28/28 [==============================] - 0s 797us/step - loss: 0.1222 -
accuracy: 0.9562
Epoch 12/15
28/28 [==============================] - 0s 796us/step - loss: 0.1143 -
accuracy: 0.9607
Epoch 13/15
28/28 [==============================] - 0s 778us/step - loss: 0.1126 -
accuracy: 0.9618
Epoch 14/15
28/28 [==============================] - 0s 748us/step - loss: 0.1083 -
accuracy: 0.9663
Epoch 15/15
28/28 [==============================] - 0s 741us/step - loss: 0.1108 -
accuracy: 0.9574
```

[52]:
```
<keras.callbacks.History at 0x2466d269d30>
```

[53]:
```
predictions = model.predict(final_test)
```

```
14/14 [==============================] - 0s 696us/step
```

[54]:
```
rounded_predictions = [round(p[0]) for p in predictions.tolist()]
```

[55]:
```
final2 = pd.DataFrame()
final2['Survived'] = rounded_predictions
final2['PassengerId'] = data['PassengerId'][len(train):]
final2.to_csv('submission2.csv');
```

1.0.10 Results

Kaggle verified accuracy: 78.71%
Leaderboard position: 1775 / 15841 as of April 1st, 2024