Jon Snow: Dead or Alive

Lakhi Charan Mahato

Abstract

- This project uses the dataset of ‘Game of Thrones’ from Kaggle. The dataset was updated last in 2016. In the world of Game of Thrones nothing is certain. The show is known for the unexpected deaths, some of which are still alive in the books.
- The question that arises in the mind of every GoT fan is ‘Can predict the life or death future of a character or to what extend are the books followed?’.
- This project uses Regression to solve the problem. The Regressions used are Linear Regression and Decision Tree Regressor.
- In this project findings turn out to be such that it becomes possible to predict the life or death of a character to a very high accuracy.

Motivation

In the world of Game of Thrones the characters have a very short life. The show is well known for the frequent character deaths and many unexpected deaths. The directors of the show do not follow the books in certain aspects, mostly the death of characters. Almost 59 character deaths have been shown in the show who are still alive in the books(http://mentalfloss.com/article/65078/dead-game-thrones-tv-characters-who-are-still-alive-books). No place is safe and nobody is safe. There is a war everywhere. Even the powerful characters have an uncertain death. No character is favourite to the directors. Nobility doesn't even matter. Women or children, no one is safe. In such a scenario it becomes very difficult to predict the death of a character which is what the goal of the project.

Dataset(s)

- Game of Thrones

This is a Kaggle dataset(https://www.kaggle.com/mylesoneill/game-of-thrones). Game of Thrones is a hit fantasy tv show based on the equally famous book series "A Song of Fire and Ice" by George RR Martin. The show is well known for its vastly complicated political landscape, large number of characters, and its frequent character deaths. The dataset contains three data sources- battles.csv, character-deaths.csv, character-predictions.csv. In this project analysis has been done on ‘character-deaths.csv’. This dataset contains the certain features of a character such as popularity, nobility, relations, etc.

Dataset(s)

The columns present are S.No', 'actual', 'pred', 'alive', 'plod', 'name', 'title', 'male', 'culture', 'dateOfBirth', 'DateoFdeath', 'mother', 'father', 'heir', 'house', 'spouse', 'book1', 'book2', 'book3', 'book4', 'book5','isAliveMother', 'isAliveFather', 'isAliveHeir', 'isAliveSpouse', 'isMarried', 'isNoble', 'age', 'numDeadRelations', 'boolDeadRelations','isPopular', 'popularity', 'isAlive‘. The ‘actual’ column contains the life or death state of the characters in the books. The ‘pred’ column contains the prediction of whether a character will live or die. The ‘alive’ column contains the probability of a character to stay alive. The ‘plod’ column contains the probability of a character’s death.

Data Preparation and Cleaning

- The ‘character-deaths.csv’ contains a lot of unuseful information like S.No, name, title, date of birth, death of death, mother, father, heir and spouse. So these columns were removed from the table.
- There were a certain columns like culture, isAliveMother, isAliveFather, isAliveHeir, isAliveSpouse, age, which have almost **80%** of their columns having null values, so they were also removed.
- There was a column named ‘house’ which is used to generate a new column called **‘nrank’** which contains the fraction of dead members to total number of members of a particular house.

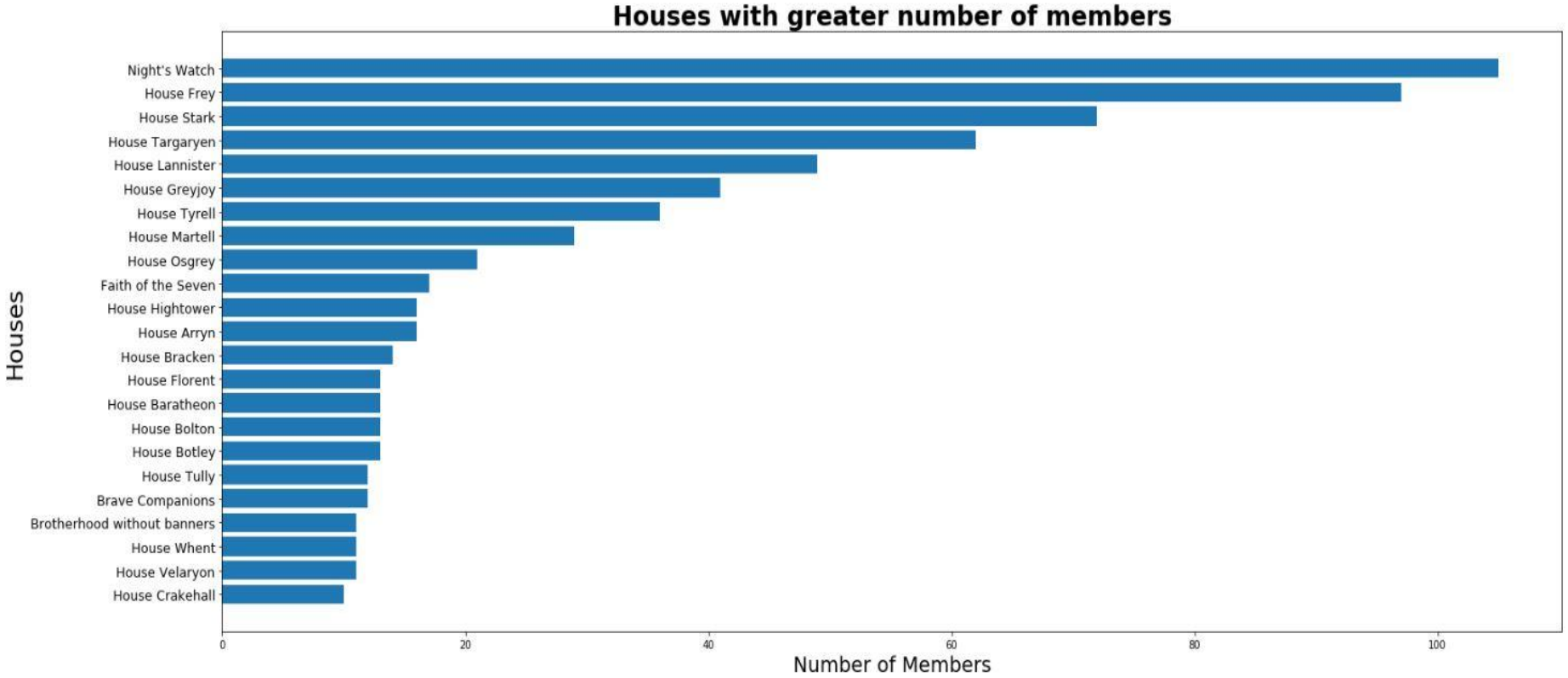
Research Question(s)

- Can we predict the death of a Game of Thrones character?
- To what extend the directors of the show follow the book ‘A song of Ice and Fire’ as far as **death** of a character is concerned?
- Which house has greater number of members?

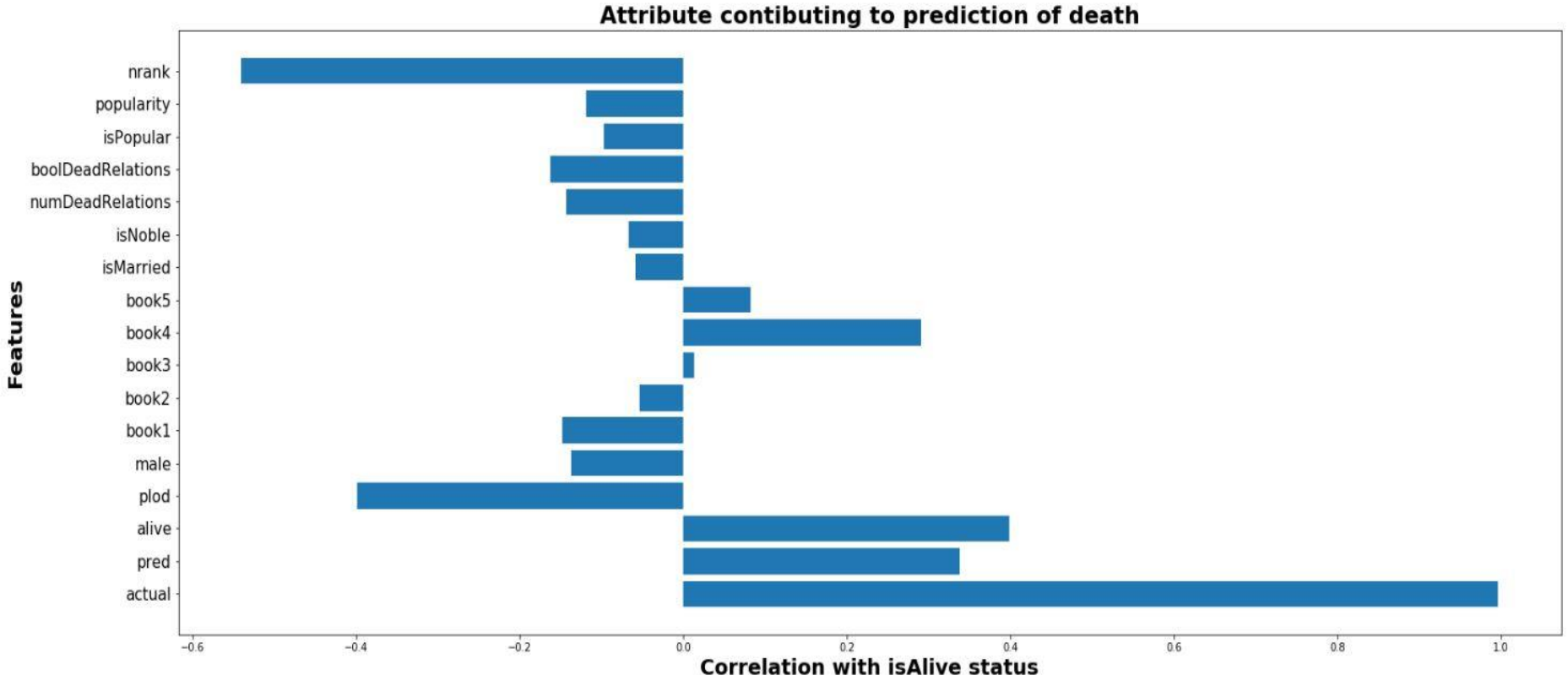
Methods

This project basically uses the two Regression models- Linear Regression and Decision Tree Regressor. This was an obvious choice because we needed to predict a numerical value 1 or 0(life and death respectively). Moreover the Root Mean Square Error(RMSE) is calculated to find deviation from the correct prediction. Two-thirds of the model was used for Training the model and the remaining one-third was used for Testing purpose.

Findings


Houses with greater number of members
Houses
Night's Watch
House Frey
House Stark
House Targaryen
House Lannister
House Greyjoy
House Tyrell
House Martell
House Osgrey
Faith of the Seven
House Hightower
House Arryn
House Bracken
House Florent
House Baratheon
House Bolton
House Botley
House Tully
Brave Companions
Brotherhood without banners
House Whent
House Velaryon
House Crakehall
0
20
40
60
80
100
Number of Members

Findings


Attribute contibuting to prediction of death
nrank
popularity
isPopular
boolDeadRelations
numDeadRelations
isNoble
isMarried
book5
book4
book3
book2
book1
male
plod
alive
pred
actual
Features
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Correlation with isAlive status

Findings

- ✓ Houses with most number of members in descending order:
- Night’s watch
- House Frey
- House Stark
- House Targaryen
- House Lannister
- House Greyjoy
- House Tyrell
- House Martell
- House Osgrey
- Faith of the Seven
- House High Tower
- House Arryn
- House Bracken
- House Florent
- House Baratheon
- House Bolton
- House Botley
- House Tully
- Brave Companions
- Brotherhood without Banners

Findings

- ✓ Attributes having a significant correlation with the ‘isAlive’ column which represents the current life or death state of a person:
- ❑ actual(very high)[Life or death state of a character in the book]
- ❑ nrank
- ❑ book4
- ❑ plod
- ❑ alive
- ❑ pred

Findings

- ✓ Predicting the death of a character taking ‘actual’ column, which shows the life or death state of the character in the book, into consideration:
- Linear Regression –
  RMSE=0.0056416749160699 7
- Decision Tree Regression -
  RMSE=0.0
- ✓ Without taking ‘actual’ column into consideration:
- Linear Regression –
  RMSE=0.4016820256109475 7
- Decision Tree Regression -
  RMSE=0.4016820256109475 7

Findings

- ✓ The ‘actual’ column if removed there comes a minute error of 0.402 in both Linear Regression and Decision Tree Regression.
- ✓ When the ‘actual’ column is taken into consideration, the RMSE for Linear Regression comes out to be very low and for Decision Tree Regression it comes out to be 0.0 which is almost perfect. Thus it is possible to predict the death of a character using the book to a very high accuracy.
- ✓ From the low value of RMSE we can say that although the directors of show of the show do not follow the book sometimes but they follow a specific pattern to kill the characters.

Limitations

From the Regression model we can calculate the value of future state of a character to a great extent. However when asked about a certain important character of the show, there is a possibility of error. Moreover if the character has certain missing information we cannot tell its future. If a certain character which is not present in the book is introduced we cannot tell about his future.

Conclusions

- We can predict the death of a character to a very high accuracy.
- The directors of the show mostly follow the book for the death of a character but they do make changes sometimes.
- Night’s Watch, House Frey and House Stark are top 3 houses with most number of members.

Acknowledgements

I would like to thank the instructors- Ilkay Altintas and Leo Porter for their teachings throughout the course. That was a great help for me in my project. Moreover the jupyter notebooks served as a great reference for my project. I Googled the things which I did not know or whenever I got stuck.

References

I took the help of certain websites whenever I got stuck or wanted to learn something I did not know:

- https://www.kaggle.com/mylesoneill/game-of-thrones
- http://mentalfloss.com/article/65078/dead-game-thrones-tv-characters-who-are-still-alive-books

In [1]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [2]:
```
data = pd.read_csv('character-predictions.csv')
data.shape
```

Out[2]:
```
(1946, 33)
```

In [3]:
```
data.columns
```

Out[3]:
```
Index(['S.No', 'actual', 'pred', 'alive', 'plod', 'name', 'title', 'male',
       'culture', 'dateOfBirth', 'DateoFdeath', 'mother', 'father', 'heir',
       'house', 'spouse', 'book1', 'book2', 'book3', 'book4', 'book5',
       'isAliveMother', 'isAliveFather', 'isAliveHeir', 'isAliveSpouse',
       'isMarried', 'isNoble', 'age', 'numDeadRelations', 'boolDeadRelation
s',
       'isPopular', 'popularity', 'isAlive'],
      dtype='object')
```

In [4]:
```
data['pred'].unique()
```

Out[4]:
```
array([0, 1], dtype=int64)
```

In [5]:
```
data['isAlive'].unique()
```

Out[5]:
```
array([0, 1], dtype=int64)
```

Data Preparation and Cleaning Phase:

In [6]:
```
#Deleting columns which would be of no help in predicting the target variable
 'isAlive'
del data['S.No']
del data['name']
del data['title']
del data['dateOfBirth']
del data['DateoFdeath']
del data['mother']
del data['father']
del data['heir']
del data['spouse']
```

In [7]:
```
#This will give number of NaN values in every column.
data.isnull().sum(axis = 0)
```

Out[7]:
```
actual                 0
pred                   0
alive                  0
plod                   0
male                   0
culture             1269
house                427
book1                  0
book2                  0
book3                  0
book4                  0
book5                  0
isAliveMother       1925
isAliveFather       1920
isAliveHeir         1923
isAliveSpouse       1670
isMarried              0
isNoble                0
age                 1513
numDeadRelations       0
boolDeadRelations      0
isPopular              0
popularity             0
isAlive                0
dtype: int64
```

In [8]:
```
data.isnull().sum(axis = 1).count()
```

Out[8]:
```
1946
```

In [9]:
```
#Deleting columns that contains a lot number of nullvalues which be decrease t
he dataset to smallsize.
del data['culture']
del data['isAliveMother']
del data['isAliveFather']
del data['isAliveHeir']
del data['isAliveSpouse']
del data['age']
```

In [10]:
```
data.isnull().sum(axis = 0)
```

Out[10]:
```
actual                0
pred                  0
alive                 0
plod                  0
male                  0
house               427
book1                 0
book2                 0
book3                 0
book4                 0
book5                 0
isMarried             0
isNoble               0
numDeadRelations      0
boolDeadRelations     0
isPopular             0
popularity            0
isAlive               0
dtype: int64
```

In [11]:
```
data['house'].unique()
```

Out[11]:
```
array([nan, 'House Frey', 'House Swyft', 'House Arryn', 'House Santagar',
       'House Targaryen', 'House Osgrey', "Night's Watch", 'House Humble',
       'House Wylde', 'House Wode', 'House Fell',
       'Brotherhood Without Banners', 'House Webber', 'House Greyjoy',
       'House Stark', 'House Waynwood', 'House Dayne', 'House Manderly',
       'House Farwynd of the Lonely Light', 'Happy Port',
       'House of Loraq', 'Kingswood Brotherhood', 'House Botley',
       'Burned Men', 'House Velaryon', 'House Tallhart', 'House Tyrell',
       'House Blackwood', 'House Blackfyre', 'wildling',
       'Kingdom of the Three Daughters',
       'House Royce of the Gates of the Moon', 'House Nayland',
       "House Vance of Wayfarer's Rest", 'House Rowan', 'House Farrow',
       'House Lonmouth', 'House Reyne', 'House Ashford', 'House Brax',
       'House Paege', 'House Hollard', 'House Tarth', 'House Ryswell',
       'House Lannister', 'House Crakehall', 'House Darklyn',
       'House Westerling', 'House Weaver', 'House Connington',
       'House Stout', 'House Harclay', 'House Drumm',
       'House Lannister of Lannisport', 'House Crabb', 'Drowned men',
       'Stormcrows', 'Antler Men', 'House Spicer', 'House Staunton',
       'House Stonetree', 'House Leygood', 'House Bolton', 'House Sparr',
       'House Fossoway of Cider Hall', 'House Varner', 'House Tarly',
       'Peach', 'House Lynderly', 'House Ironmaker', 'House Martell',
       'House Clegane', 'House Costayne', 'House Heddle', 'Stone Crows',
       'House Ryger', 'Sea watch', 'Second Sons', 'Moon Brothers',
       'House Baelish', 'Faceless Men', 'Brave Companions',
       'House Frey of Riverrun', 'House Stokeworth', 'House Hightower',
       'Band of Nine', 'House Oakheart', 'House Sharp', 'House Hunt',
       'House Bracken', 'House Whent', 'House Dalt', 'House Charlton',
       'House Butterwell', 'House Qorgyle', 'House Ambrose',
       "Alchemists' Guild", 'House Bushy', 'House Blacktyde',
       'House Beesbury', 'House Baratheon', 'House Bulwer',
       'House Cafferen', 'House Smallwood', 'House Payne',
       'House Buckler', 'House Tully', 'Black Ears', 'House Caswell',
       'House Conklyn', 'House Greenhill', 'House Karstark',
       'House Redfort', 'House Baratheon of Dragonstone',
       'House Deddings', 'House Slynt', 'House Plumm', 'House Redwyne',
       'House Manwoody', 'House Harlaw', 'House Risley',
       'Faith of the Seven', 'Pureborn', 'House Codd', 'House Willum',
       'House of Galare', "R'hllor", 'House Morrigen', 'House Corbray',
       'House Gaunt', 'House Goodbrother', 'Kingsguard', 'House Grafton',
       'House Hornwood', 'House Grandison', 'House Sawyer',
       'House Uffering', 'House Bolling', 'House Clifton',
       'House Blackberry', "Chataya's brothel", 'House Norcross',
       'House Mallister', 'House Fossoway of New Barrel', 'House Swann',
       'House Cupps', 'House Penny', 'House Penrose', 'House Umber',
       'House Vance of Atranta', 'Brotherhood without banners',
       'House Royce', 'House Tawney', 'Windblown', 'House Piper',
       'House Serry', 'House Lorch', 'House Lefford', 'House Strong',
       'House Roote', 'House Hardy', 'Queensguard', 'House Meadows',
       'House Dondarrion', 'House Lothston', 'House Yew',
       'House Mullendore', 'House Florent', 'House Mertyns',
       'House Boggs', 'House Woods', 'House Inchfield', 'House Blackmont',
       'House Pemford', 'House Kenning of Harlaw', 'House Shepherd',
       'House Estermont', 'House Wynch', 'House Staedmon',
       'House Seaworth', "City Watch of King's Landing", 'House Cockshaw',
       'House Graceford', 'House Stackspear', 'House Haigh', 'House Jast',
       'House Farring', 'House Celtigar', 'House of Ghazeen',
```

```
'House Byrch', 'House Hawick', 'House Belmore', 'House Broom',
'House Brune of Brownhollow', 'House Cassel', 'House Rosby',
'House Harlaw of Harridan Hill', 'House Cuy', 'House Caron',
'House Cerwyn', 'House Vaith', 'House Vance',
'House Shett of Gull Tower', 'House Vypren', 'House Marbrand',
'House Hardyng', 'House Allyrion', 'House Locke',
'House Bar Emmon', 'House Mooton', 'House Norridge',
'House Hunter', 'House Hayford', 'House Glover',
'House Brune of the Dyre Den', 'House Prester', 'House Fowler',
'House Goodbrook', 'House Greenfield', 'House Gower', 'Blacks',
'House Hewett', 'House Borrell', 'House Peake', 'Citadel',
'Wise Masters', 'Good Masters', 'House Sunglass', 'House Grimm',
'The Citadel', 'House Mollen', 'House Hoare', 'House Rambton',
'House Harlaw of the Tower of Glimmering', 'House Wull',
'House Wagstaff', 'House Vyrwel', 'House Darry', 'House Lydden',
'House Bettley', 'House Myre', 'House Mormont', 'House Reed',
'House Kenning of Kayce', 'House of Pahl', 'House Turnberry',
'House Blackbar', 'House Woolfield', 'House Fossoway',
'House Mallery', 'House Chyttering',
"House Flint of Widow's Watch", 'House Lychester', 'House Vikary',
'House Selmy', 'House Volmark', 'House Merlyn', 'House Crane',
'House Sarsfield', 'House of Merreq', 'House Chester',
'House Jordayne', 'Iron Bank of Braavos',
'House Goodbrother of Shatterstone', 'House Toland',
"House Baratheon of King's Landing", 'House Yronwood',
'House Kettleblack', 'House Wythers', 'House Norrey',
'House Foote', 'House Chelsted', 'House Banefort', 'House Ball',
'House Cox', 'House Ruttiger', 'House Estren', 'House Rykker',
'House Longwaters', 'House Thorne', 'House Moreland',
'House Flint', 'House Hogg', 'House Longthorpe', 'House Coldwater',
'House Leek', 'House Farman', 'House Harlaw of Harlaw Hall',
'House Templeton', 'House Wells', 'House Toyne', 'House Liddle',
'House Gargalen', 'House Mudd', 'House Farwynd',
'House Sunderland', 'House Hetherspoon', 'House Uller',
'House Wayn', 'House Tollett', 'House Poole', 'Maesters',
'House Blanetree', 'House Blount', 'Golden Company',
'Company of the Cat', 'House Suggs', 'Khal',
'House Nymeros Martell', 'House Drinkwater',
'House Harlaw of Grey Garden', 'Summer Islands', 'House Condon',
'House Lannister of Casterly Rock', 'House Moore', 'House Trant',
'House Yarwyck', 'Undying Ones', 'House Stonehouse',
'House Bolton of the Dreadfort', 'Thenn', 'House Hasty',
'House Cole', 'Graces', 'House Tyrell of Brightwater Keep',
'House Dayne of High Hermitage', 'Unsullied', 'House Strickland',
'House Bywater', 'House Massey', 'Brotherhood without Banners',
'House Rhysling', 'House Potter', 'House Horpe', 'House of Kandaq',
'House of Reznak', 'House Peckledon', 'House Dustin',
'Mance Rayder', 'House Egen', 'House Merryweather', 'House Errol',
'Thirteen', 'House Erenford', 'House Grell',
'brotherhood without banners', 'Three-eyed crow'], dtype=object)
```

In [12]:
```
ho=['House Frey', 'House Swyft', 'House Arryn', 'House Santagar',
       'House Targaryen', 'House Osgrey', "Night's Watch", 'House Humble',
       'House Wylde', 'House Wode', 'House Fell',
       'Brotherhood Without Banners', 'House Webber', 'House Greyjoy',
       'House Stark', 'House Waynwood', 'House Dayne', 'House Manderly',
       'House Farwynd of the Lonely Light', 'Happy Port',
       'House of Loraq', 'Kingswood Brotherhood', 'House Botley',
       'Burned Men', 'House Velaryon', 'House Tallhart', 'House Tyrell',
       'House Blackwood', 'House Blackfyre', 'wildling',
       'Kingdom of the Three Daughters',
       'House Royce of the Gates of the Moon', 'House Nayland',
       "House Vance of Wayfarer's Rest", 'House Rowan', 'House Farrow',
       'House Lonmouth', 'House Reyne', 'House Ashford', 'House Brax',
       'House Paege', 'House Hollard', 'House Tarth', 'House Ryswell',
       'House Lannister', 'House Crakehall', 'House Darklyn',
       'House Westerling', 'House Weaver', 'House Connington',
       'House Stout', 'House Harclay', 'House Drumm',
       'House Lannister of Lannisport', 'House Crabb', 'Drowned men',
       'Stormcrows', 'Antler Men', 'House Spicer', 'House Staunton',
       'House Stonetree', 'House Leygood', 'House Bolton', 'House Sparr',
       'House Fossoway of Cider Hall', 'House Varner', 'House Tarly',
       'Peach', 'House Lynderly', 'House Ironmaker', 'House Martell',
       'House Clegane', 'House Costayne', 'House Heddle', 'Stone Crows',
       'House Ryger', 'Sea watch', 'Second Sons', 'Moon Brothers',
       'House Baelish', 'Faceless Men', 'Brave Companions',
       'House Frey of Riverrun', 'House Stokeworth', 'House Hightower',
       'Band of Nine', 'House Oakheart', 'House Sharp', 'House Hunt',
       'House Bracken', 'House Whent', 'House Dalt', 'House Charlton',
       'House Butterwell', 'House Qorgyle', 'House Ambrose',
       "Alchemists' Guild", 'House Bushy', 'House Blacktyde',
       'House Beesbury', 'House Baratheon', 'House Bulwer',
       'House Cafferen', 'House Smallwood', 'House Payne',
       'House Buckler', 'House Tully', 'Black Ears', 'House Caswell',
       'House Conklyn', 'House Greenhill', 'House Karstark',
       'House Redfort', 'House Baratheon of Dragonstone',
       'House Deddings', 'House Slynt', 'House Plumm', 'House Redwyne',
       'House Manwoody', 'House Harlaw', 'House Risley',
       'Faith of the Seven', 'Pureborn', 'House Codd', 'House Willum',
       'House of Galare', "R'hllor", 'House Morrigen', 'House Corbray',
       'House Gaunt', 'House Goodbrother', 'Kingsguard', 'House Grafton',
       'House Hornwood', 'House Grandison', 'House Sawyer',
       'House Uffering', 'House Bolling', 'House Clifton',
       'House Blackberry', "Chataya's brothel", 'House Norcross',
       'House Mallister', 'House Fossoway of New Barrel', 'House Swann',
       'House Cupps', 'House Penny', 'House Penrose', 'House Umber',
       'House Vance of Atranta', 'Brotherhood without banners',
       'House Royce', 'House Tawney', 'Windblown', 'House Piper',
       'House Serry', 'House Lorch', 'House Lefford', 'House Strong',
       'House Roote', 'House Hardy', 'Queensguard', 'House Meadows',
       'House Dondarrion', 'House Lothston', 'House Yew',
       'House Mullendore', 'House Florent', 'House Mertyns',
       'House Boggs', 'House Woods', 'House Inchfield', 'House Blackmont',
       'House Pemford', 'House Kenning of Harlaw', 'House Shepherd',
       'House Estermont', 'House Wynch', 'House Staedmon',
       'House Seaworth', "City Watch of King's Landing", 'House Cockshaw',
       'House Graceford', 'House Stackspear', 'House Haigh', 'House Jast',
```

```
       'House Farring', 'House Celtigar', 'House of Ghazeen',
       'House Byrch', 'House Hawick', 'House Belmore', 'House Broom',
       'House Brune of Brownhollow', 'House Cassel', 'House Rosby',
       'House Harlaw of Harridan Hill', 'House Cuy', 'House Caron',
       'House Cerwyn', 'House Vaith', 'House Vance',
       'House Shett of Gull Tower', 'House Vypren', 'House Marbrand',
       'House Hardyng', 'House Allyrion', 'House Locke',
       'House Bar Emmon', 'House Mooton', 'House Norridge',
       'House Hunter', 'House Hayford', 'House Glover',
       'House Brune of the Dyre Den', 'House Prester', 'House Fowler',
       'House Goodbrook', 'House Greenfield', 'House Gower', 'Blacks',
       'House Hewett', 'House Borrell', 'House Peake', 'Citadel',
       'Wise Masters', 'Good Masters', 'House Sunglass', 'House Grimm',
       'The Citadel', 'House Mollen', 'House Hoare', 'House Rambton',
       'House Harlaw of the Tower of Glimmering', 'House Wull',
       'House Wagstaff', 'House Vyrwel', 'House Darry', 'House Lydden',
       'House Bettley', 'House Myre', 'House Mormont', 'House Reed',
       'House Kenning of Kayce', 'House of Pahl', 'House Turnberry',
       'House Blackbar', 'House Woolfield', 'House Fossoway',
       'House Mallery', 'House Chyttering',
       "House Flint of Widow's Watch", 'House Lychester', 'House Vikary',
       'House Selmy', 'House Volmark', 'House Merlyn', 'House Crane',
       'House Sarsfield', 'House of Merreq', 'House Chester',
       'House Jordayne', 'Iron Bank of Braavos',
       'House Goodbrother of Shatterstone', 'House Toland',
       "House Baratheon of King's Landing", 'House Yronwood',
       'House Kettleblack', 'House Wythers', 'House Norrey',
       'House Foote', 'House Chelsted', 'House Banefort', 'House Ball',
       'House Cox', 'House Ruttiger', 'House Estren', 'House Rykker',
       'House Longwaters', 'House Thorne', 'House Moreland',
       'House Flint', 'House Hogg', 'House Longthorpe', 'House Coldwater',
       'House Leek', 'House Farman', 'House Harlaw of Harlaw Hall',
       'House Templeton', 'House Wells', 'House Toyne', 'House Liddle',
       'House Gargalen', 'House Mudd', 'House Farwynd',
       'House Sunderland', 'House Hetherspoon', 'House Uller',
       'House Wayn', 'House Tollett', 'House Poole', 'Maesters',
       'House Blanetree', 'House Blount', 'Golden Company',
       'Company of the Cat', 'House Suggs', 'Khal',
       'House Nymeros Martell', 'House Drinkwater',
       'House Harlaw of Grey Garden', 'Summer Islands', 'House Condon',
       'House Lannister of Casterly Rock', 'House Moore', 'House Trant',
       'House Yarwyck', 'Undying Ones', 'House Stonehouse',
       'House Bolton of the Dreadfort', 'Thenn', 'House Hasty',
       'House Cole', 'Graces', 'House Tyrell of Brightwater Keep',
       'House Dayne of High Hermitage', 'Unsullied', 'House Strickland',
       'House Bywater', 'House Massey', 'Brotherhood without Banners',
       'House Rhysling', 'House Potter', 'House Horpe', 'House of Kandaq',
       'House of Reznak', 'House Peckledon', 'House Dustin',
       'Mance Rayder', 'House Egen', 'House Merryweather', 'House Errol',
       'Thirteen', 'House Erenford', 'House Grell',
       'brotherhood without banners', 'Three-eyed crow']
```

In [13]:
```
pro=[]
proho=[]
coun=[]
death=[]
```

In [14]:
```
#Deleting rows with null  values
data=data.dropna()
```

In [15]:
```
for i in ho:
    coun.append(len(data[data['house']==i]))
    if(len(data[data['house']==i])>=10):
        pro.append(len(data[data['house']==i]))
        proho.append(i)
        print(i,"=",len(data[data['house']==i]))
```

```
House Frey = 97
House Arryn = 16
House Targaryen = 62
House Osgrey = 21
Night's Watch = 105
House Greyjoy = 41
House Stark = 72
House Botley = 13
House Velaryon = 11
House Tyrell = 36
House Lannister = 49
House Crakehall = 10
House Bolton = 13
House Martell = 29
Brave Companions = 12
House Hightower = 16
House Bracken = 14
House Whent = 11
House Baratheon = 13
House Tully = 12
Faith of the Seven = 17
Brotherhood without banners = 11
House Florent = 13
```

In [16]:
```
keydict = dict(zip(proho, pro))
proho.sort(key=keydict.get)
```

In [17]:
```
keydict = dict(zip(ho, coun))
ho.sort(key=keydict.get)
```

In [18]:
```
pro=[]
coun=[]
```

In [19]:
```
for i in proho:
        pro.append(len(data[data['house']==i]))
```

In [20]:
```
for i in ho:
        coun.append(len(data[data['house']==i]))
```

In [21]:
```
for i in ho:
    c=0
    for ind in data.index:
        if(data['isAlive'][ind]==0 and data['house'][ind]==i):
            c+=1
    death.append(c)
```

In [22]:
```
rank=[]
for i in range(0,len(ho)):
    rank.append(death[i]/coun[i])
```

In [23]:
```
keydict = dict(zip(ho,rank))
```

In [24]:
```
keydict['Three-eyed crow']
```

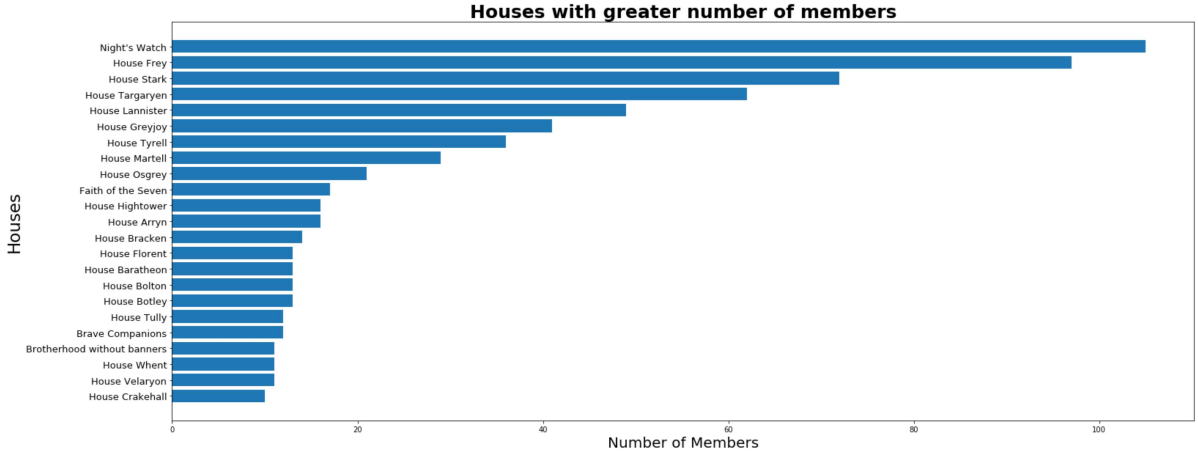
Out[24]: 0.0

In [25]:
```
list2=[]
for ind in data.index:
    list2.append(keydict[data['house'][ind]])
data=data.assign(nrank=list2)
#data['nrank']=list2
#data.to_csv('character-predictions.csv', index=False)
```

In [26]:
```
len(proho)
```

Out[26]: 23

In [27]:
```
from pylab import rcParams
rcParams['figure.figsize'] = 25, 10
index = np.arange(len(pro))
plt.barh(index, pro)
plt.ylabel('Houses', fontsize=23)
plt.xlabel('Number of Members', fontsize=20)
plt.yticks(index, proho, fontsize=13)
plt.title('Houses with greater number of members',fontweight='bold',fontsize=25);
plt.show()
```



In [28]:
```
c=0
for i in ho:
    c+=len(data[data['house']==i])
print(c)
```

```
1519
```

In [29]:
```
data.isnull().sum(axis = 0)
```

Out[29]:
```
actual              0
pred                0
alive               0
plod                0
male                0
house               0
book1               0
book2               0
book3               0
book4               0
book5               0
isMarried           0
isNoble             0
numDeadRelations    0
boolDeadRelations   0
isPopular           0
popularity          0
isAlive             0
nrank               0
dtype: int64
```

Data Analysis Phase:

In [30]:
```
#Finding Correlation of various columns with target variable 'isAlive'
data['isAlive'].corr(data['actual'])
```

Out[30]:
```
0.996606344950849
```

Note the extraordinary correlation between 'actual' and 'isAlive' columns

In [31]:
```
data['isAlive'].corr(data['book4'])
```

Out[31]:
```
0.29131821434456245
```

In [32]:
```
data['isAlive'].corr(data['nrank'])
```

Out[32]:
```
-0.54075773590686
```

In [33]:
```
data['isAlive'].corr(data['alive'])
```
Out[33]: 0.3987029544970314

In [34]:
```
data['isAlive'].corr(data['pred'])
```
Out[34]: 0.33838621900346855

In [35]:
```
data['isAlive'].corr(data['plod'])
```
Out[35]: -0.3987029544970313

In [36]:
```
data['isAlive'].corr(data['male'])
```
Out[36]: -0.13699798596736582

In [37]:
```
data['isAlive'].corr(data['book1'])
```
Out[37]: -0.14841950098158782

In [38]:
```
data['isAlive'].corr(data['book2'])
```
Out[38]: -0.05368637142613707

In [39]:
```
data['isAlive'].corr(data['book3'])
```
Out[39]: 0.013175942638259972

In [40]:
```
data['isAlive'].corr(data['book5'])
```
Out[40]: 0.08180058034313706

In [41]:
```
data['isAlive'].corr(data['isMarried'])
```
Out[41]: -0.057676046019269143

In [42]:
```
data['isAlive'].corr(data['isNoble'])
```
Out[42]: -0.06710791676461056

In [43]:
```
data['isAlive'].corr(data['numDeadRelations'])
```
Out[43]: -0.14254266994147907

In [44]:
```
data['isAlive'].corr(data['boolDeadRelations'])
```
Out[44]: -0.16291627626319918

In [45]:
```
data['isAlive'].corr(data['popularity'])
```
Out[45]: -0.1184881409821493

In [46]:
```
data['isAlive'].corr(data['isPopular'])
```

Out[46]:
```
-0.0970680564370428
```

In [47]:
```
data.columns
```

Out[47]:
```
Index(['actual', 'pred', 'alive', 'plod', 'male', 'house', 'book1', 'book2',
       'book3', 'book4', 'book5', 'isMarried', 'isNoble', 'numDeadRelations',
       'boolDeadRelations', 'isPopular', 'popularity', 'isAlive', 'nrank'],
      dtype='object')
```

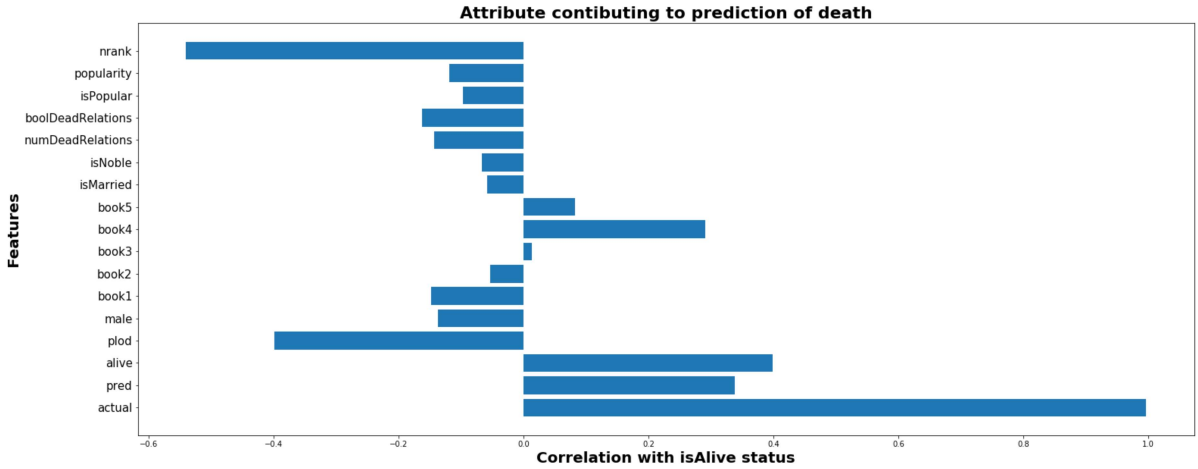
Declaring Columns which are to be Used as Features

In [48]:
```
features=['actual','pred', 'alive', 'plod', 'male', 'book1', 'book2',
       'book3', 'book4', 'book5', 'isMarried', 'isNoble', 'numDeadRelations',
       'boolDeadRelations', 'isPopular', 'popularity','nrank']
m=[]
```

In [49]:
```
for i in features:
    m.append(data['isAlive'].corr(data[i]))
```

Data Visualization

In [50]:
```
from pylab import rcParams
rcParams['figure.figsize'] = 25, 10
index = np.arange(len(features))
plt.barh(index, m)
plt.ylabel('Features', fontsize=20,fontweight='bold')
plt.xlabel('Correlation with isAlive status', fontsize=20,fontweight='bold')
plt.yticks(index, features, fontsize=15)
plt.title('Attribute contibuting to prediction of death',fontweight='bold',fon
tsize=22)
plt.show()
```



Data Analysis to predict target value using Regression Models

Specifying the Prediction Target

In [51]:
```
target = ['isAlive']
```

Extracting Features and Target ('isAlive') Values into Separate Dataframes

In [52]:
```
X = data[features]
```

In [53]:
```
y = data[target]
```

In [54]:
```
X.iloc[2]
```

Out[54]:
```
actual               0.000000
pred                 0.000000
alive                0.076000
plod                 0.924000
male                 0.000000
book1                0.000000
book2                0.000000
book3                0.000000
book4                0.000000
book5                0.000000
isMarried            1.000000
isNoble              1.000000
numDeadRelations     0.000000
boolDeadRelations    0.000000
isPopular            0.000000
popularity           0.183946
nrank                0.375000
Name: 3, dtype: float64
```

In [55]:
```
y
```

Out[55]:

	isAlive
1	1
2	1
3	0
4	1
6	0
9	1
10	0
13	1
14	1
15	1
17	1
19	1
20	1
21	1
22	1
24	1
27	1
29	0
30	0
32	1
33	1
35	1
36	1
37	1
38	1
41	0
46	1
47	1
48	1
49	1
...	...
1912	0
1913	0
1914	1
1915	0

	isAlive
1916	1
1917	1
1918	0
1920	1
1921	0
1922	1
1923	1
1924	1
1925	1
1926	1
1927	1
1928	1
1929	0
1930	1
1931	1
1932	1
1934	0
1935	1
1936	1
1938	1
1939	1
1940	0
1941	0
1942	0
1944	1
1945	0

1519 rows × 1 columns

Extracting Features and Target ('overall_rating') Values into Separate Dataframes

In [56]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rand
om_state=324)
```

Linear Regression: Fitting a model to the training set

In [57]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[57]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Performing Prediction using Linear Regression Model

In [58]:
```
y_prediction = regressor.predict(X_test)
y_prediction
```

Out[58]:
```
array([[ 9.91807606e-01],
       [ 1.00165476e+00],
       [ 2.88166626e-03],
       [ 9.99263623e-01],
       [ 1.00126624e+00],
       [ 9.95953157e-01],
       [ 1.00813689e+00],
       [ 9.86073270e-01],
       [ 9.94374107e-01],
       [ 1.00111190e+00],
       [-2.27406600e-03],
       [ 1.00818839e+00],
       [ 9.88916167e-01],
       [ 9.92150531e-01],
       [ 1.00263065e+00],
       [ 1.00099441e+00],
       [ 1.00196107e+00],
       [ 1.00265923e+00],
       [ 1.00878784e+00],
       [ 1.00535182e+00],
       [ 1.00577583e+00],
       [ 9.93596571e-01],
       [ 9.93187442e-01],
       [-2.90117658e-03],
       [ 9.99151539e-01],
       [ 9.88629117e-01],
       [ 9.94777896e-01],
       [ 9.94617817e-01],
       [ 7.44021676e-03],
       [ 9.97189433e-01],
       [ 5.17920116e-05],
       [ 9.93604737e-01],
       [ 1.00227101e+00],
       [ 1.00081552e+00],
       [ 9.95264394e-01],
       [ 1.00628723e+00],
       [ 1.00261651e+00],
       [ 9.97333496e-01],
       [ 9.95209674e-01],
       [ 7.59345920e-03],
       [ 1.00225926e+00],
       [ 1.00115577e+00],
       [ 9.89984867e-01],
       [ 9.96665582e-01],
       [ 1.50353643e-02],
       [ 1.00894882e+00],
       [ 1.00495520e+00],
       [ 9.85801340e-01],
       [ 1.00320690e+00],
       [ 1.00499094e+00],
       [ 1.00083493e+00],
       [ 9.95765374e-01],
       [ 4.26402196e-03],
       [ 2.07958934e-03],
       [ 9.93521870e-01],
       [ 9.89022249e-01],
       [ 1.00149728e+00],
```

```
[ 9.95673687e-01],
[ 1.01419832e+00],
[ 1.00235411e+00],
[ 9.88307350e-01],
[ 3.00138562e-03],
[ 9.95403334e-01],
[ 1.00081953e+00],
[ 9.97033557e-01],
[ 1.00761363e+00],
[-1.02098125e-04],
[ 1.00151994e+00],
[ 6.18644671e-03],
[-2.03241905e-03],
[-1.00520343e-03],
[ 1.00177455e+00],
[ 9.92027618e-01],
[ 1.00658438e+00],
[ 1.00248585e+00],
[ 1.00219121e+00],
[-5.03543006e-03],
[ 9.91866490e-01],
[ 9.91880301e-01],
[ 9.84599070e-01],
[ 1.00178078e+00],
[ 9.94642024e-01],
[ 1.01336057e+00],
[ 9.94742792e-01],
[ 5.69427202e-03],
[ 1.00011587e+00],
[ 1.00189016e+00],
[ 9.93997999e-01],
[ 1.00063961e+00],
[-1.86346040e-04],
[ 9.95927642e-01],
[ 1.00718098e+00],
[ 9.98885092e-01],
[ 1.00234368e+00],
[ 3.40941986e-03],
[ 9.95933657e-01],
[ 9.92941106e-01],
[ 1.00743359e+00],
[ 9.94695027e-01],
[ 1.00344482e+00],
[ 2.44263711e-03],
[ 1.39949714e-03],
[ 1.00046203e+00],
[ 1.00224980e+00],
[ 9.98185806e-01],
[ 9.99937671e-01],
[ 9.92348978e-01],
[ 2.36160369e-03],
[ 9.98061412e-01],
[ 9.98184221e-01],
[ 4.11982247e-03],
[ 6.42365921e-03],
[ 9.95181885e-01],
[ 1.00841626e+00],
```

```
       [ 9.95733463e-01],
       [ 9.95067443e-01],
       [-9.95056215e-04],
       [ 1.00138817e+00],
       [ 9.97861758e-01],
       [ 1.00038164e+00],
       [ 9.96519569e-01],
       [ 1.00366702e+00],
       [-1.86020647e-03],
       [ 9.95806669e-01],
       [ 1.00484919e+00],
       [ 1.00296890e+00],
       [-7.26888146e-03],
       [ 1.00455665e+00],
       [ 9.91224429e-01],
       [ 1.03749068e-02],
       [ 1.16900159e-03],
       [-3.80561708e-03],
       [ 1.00035446e+00],
       [ 1.00741084e+00],
       [ 9.95983826e-01],
       [ 9.91870608e-01],
       [ 1.00188673e+00],
       [ 9.95531015e-01],
       [ 7.44157556e-04],
       [ 4.72362624e-03],
       [ 1.40360006e-02],
       [ 9.95080210e-01],
       [ 9.94963403e-01],
       [ 9.98534742e-01],
       [ 9.92011165e-01],
       [ 9.97427240e-01],
       [ 9.97754897e-01],
       [ 9.93134694e-01],
       [ 1.00169702e+00],
       [ 1.01008982e+00],
       [ 1.00242686e+00],
       [-4.80248913e-03],
       [ 9.98936298e-01],
       [ 1.00047314e+00],
       [ 9.99583119e-01],
       [ 2.36666948e-03],
       [ 3.42250266e-03],
       [ 8.13116252e-03],
       [ 1.00899225e+00],
       [ 9.92008501e-01],
       [ 9.97462229e-01],
       [ 9.97482422e-01],
       [ 1.00348545e+00],
       [ 1.00300159e+00],
       [ 1.00005037e+00],
       [ 9.90926951e-01],
       [ 9.98304890e-01],
       [ 9.97721593e-01],
       [ 7.26831415e-04],
       [ 9.98600792e-01],
       [ 3.62173068e-03],
```

```
       [ 1.00166625e+00],
       [ 9.89784016e-01],
       [ 2.55640419e-03],
       [ 9.89017852e-03],
       [ 9.99005565e-01],
       [ 9.86987050e-01],
       [ 9.85799476e-01],
       [-7.75284755e-03],
       [ 9.94189636e-01],
       [ 9.98927557e-01],
       [ 9.96270658e-01],
       [ 9.91838985e-01],
       [ 1.00251686e+00],
       [ 1.00165737e+00],
       [ 9.97392942e-01],
       [ 9.98922479e-01],
       [ 9.93615231e-01],
       [ 9.86073270e-01],
       [ 1.00194874e+00],
       [-4.16316838e-04],
       [ 9.96014446e-01],
       [ 9.89834818e-01],
       [ 9.94859586e-01],
       [ 1.00004403e+00],
       [ 1.00298101e+00],
       [ 1.00290047e+00],
       [ 1.00221970e+00],
       [ 1.03848715e-03],
       [ 9.96636670e-01],
       [ 9.99006705e-01],
       [ 1.00064193e+00],
       [ 1.00002162e+00],
       [ 1.00569951e+00],
       [ 1.00009483e+00],
       [-6.03381259e-03],
       [ 9.96279474e-01],
       [ 9.97111064e-01],
       [ 9.92122343e-01],
       [ 1.33297454e-03],
       [ 9.95647586e-01],
       [ 1.00105389e+00],
       [ 9.84723747e-01],
       [ 5.63482394e-03],
       [-1.01193161e-02],
       [ 1.00373016e+00],
       [ 9.98835650e-01],
       [ 1.00223249e+00],
       [ 1.00569999e+00],
       [ 9.95972953e-01],
       [ 1.00644167e+00],
       [ 9.98627769e-01],
       [ 1.00819962e+00],
       [ 8.04465731e-03],
       [ 9.99450587e-01],
       [ 9.98287230e-01],
       [ 9.97819928e-01],
       [ 1.00082701e+00],
```

```
[ 1.00459907e+00],
[ 1.00192147e+00],
[ 5.23815235e-04],
[ 7.64413430e-03],
[ 4.06082359e-03],
[ 1.00194913e+00],
[-4.55617880e-03],
[ 9.95257911e-01],
[ 9.97793602e-01],
[ 9.94618258e-01],
[ 3.07643166e-03],
[ 9.95817191e-01],
[ 1.00194913e+00],
[ 9.93771696e-01],
[ 9.92766951e-01],
[ 1.06638562e-02],
[ 4.00858346e-03],
[ 9.97524929e-01],
[-5.78888646e-04],
[ 9.95816679e-01],
[ 5.15200031e-03],
[ 9.98742098e-01],
[ 9.96156145e-01],
[ 3.89065025e-03],
[ 1.00060733e+00],
[ 1.13538646e-02],
[ 9.93681625e-01],
[ 1.00881196e+00],
[ 9.97176500e-01],
[ 9.99548868e-01],
[-3.92323827e-03],
[-4.98782025e-03],
[ 9.96927678e-01],
[ 1.00472076e+00],
[ 8.24115881e-03],
[ 1.00307774e+00],
[ 9.95803387e-01],
[ 9.93588948e-01],
[ 9.84758862e-01],
[ 9.95906172e-01],
[ 1.00282194e+00],
[ 9.89208150e-01],
[ 1.00064793e+00],
[ 9.92325456e-01],
[ 9.97946323e-01],
[ 1.00174141e+00],
[ 9.97744284e-01],
[ 1.00002519e+00],
[ 9.96101618e-01],
[ 1.00593067e+00],
[ 1.00321451e+00],
[ 1.00109351e+00],
[ 1.00901713e+00],
[-9.49875569e-05],
[ 7.27151798e-03],
[ 9.97609930e-01],
[ 1.00066264e+00],
```

```
[ 1.00534650e+00],
[ 1.00060087e+00],
[ 9.99151653e-01],
[ 1.00017037e+00],
[ 1.00197328e+00],
[ 9.99312425e-01],
[ 1.04033746e-04],
[ 9.96554053e-01],
[-5.38426655e-03],
[ 1.00223866e+00],
[ 1.00096213e+00],
[ 9.22197489e-03],
[ 9.99994207e-01],
[ 9.99140843e-01],
[ 1.00163013e+00],
[ 1.17283988e-02],
[-3.99263366e-03],
[ 1.00325639e+00],
[ 9.96133472e-01],
[ 9.94694919e-01],
[ 9.93077845e-01],
[ 9.91349475e-01],
[ 3.24257099e-04],
[ 1.00238582e+00],
[ 1.14374831e-03],
[ 9.55801581e-03],
[ 1.00375588e+00],
[ 1.00236217e+00],
[ 9.92728938e-01],
[ 9.38529784e-04],
[ 6.03448674e-03],
[ 1.00122680e+00],
[ 9.96973486e-01],
[ 9.93501871e-01],
[ 9.83594258e-01],
[ 9.97267672e-01],
[ 1.00102417e+00],
[ 1.23832855e-03],
[ 9.95106993e-01],
[ 9.97174310e-01],
[ 1.70280961e-03],
[ 1.00314503e+00],
[ 1.42959079e-03],
[ 1.00350520e+00],
[ 1.00064527e+00],
[ 1.21413582e-03],
[ 9.99780441e-01],
[ 6.52705331e-03],
[ 1.09949464e-02],
[ 6.37614035e-03],
[ 3.46926362e-03],
[-4.80329209e-04],
[ 5.17512005e-03],
[ 7.04184154e-03],
[ 1.00338472e+00],
[ 9.95486600e-01],
[ 8.67056143e-03],
```

```
[ 9.94571978e-01],
[ 9.93579518e-01],
[-2.65566255e-04],
[ 9.97952490e-01],
[ 9.97475018e-01],
[ 4.78543909e-03],
[ 1.26460661e-03],
[ 9.98427584e-01],
[ 1.06872507e-02],
[ 9.99516670e-01],
[ 1.00188300e+00],
[ 9.94854259e-01],
[ 7.04478475e-03],
[ 9.93724406e-01],
[-3.36904471e-03],
[ 9.92112518e-01],
[ 1.00082408e+00],
[ 9.92541932e-01],
[ 6.81887767e-03],
[ 9.95695450e-01],
[ 9.94125139e-01],
[ 6.94552827e-04],
[ 9.95088566e-01],
[ 9.91442286e-01],
[ 9.99689979e-01],
[ 1.00253374e+00],
[-4.29451651e-03],
[ 9.95569692e-01],
[ 1.14756966e-02],
[-9.72461140e-04],
[ 9.93545100e-01],
[ 9.94126601e-01],
[ 1.00805990e+00],
[ 9.94526001e-01],
[ 9.87545590e-01],
[ 1.00735973e+00],
[ 1.70385766e-03],
[ 9.86778823e-01],
[ 1.00401822e+00],
[ 9.96288400e-01],
[ 4.12746886e-03],
[ 9.98661754e-01],
[ 4.09439816e-03],
[ 9.98249702e-01],
[ 9.98237924e-01],
[ 9.90392013e-01],
[ 1.00841623e+00],
[ 1.00444323e+00],
[ 1.00375810e+00],
[ 1.00083455e+00],
[ 3.06700459e-03],
[ 1.00016003e+00],
[ 6.51121577e-04],
[ 1.00099768e+00],
[ 9.95418749e-01],
[ 1.00065476e+00],
[ 1.00202104e+00],
```

```
[ 9.95867984e-01],
[-3.66805442e-03],
[-6.10639072e-03],
[ 3.42250266e-03],
[ 1.00404837e+00],
[ 9.97177726e-01],
[ 1.00489876e+00],
[ 1.00279793e+00],
[ 1.00195903e+00],
[ 9.91218858e-01],
[ 9.94977079e-01],
[ 9.97255921e-01],
[ 1.00796860e+00],
[ 9.99509251e-01],
[ 1.84573206e-02],
[ 1.01036373e+00],
[ 9.92089180e-01],
[ 9.98534230e-01],
[ 9.96088272e-01],
[ 9.93890945e-01],
[ 9.94661248e-01],
[ 7.42946738e-03],
[ 9.91457752e-01],
[ 1.00140917e+00],
[ 9.96817341e-01],
[ 9.91359462e-01],
[ 1.00168523e+00],
[ 1.43035491e-02],
[ 3.90610707e-03],
[ 9.96767595e-01],
[ 1.00093426e+00],
[ 1.00546736e+00],
[ 9.92971060e-01],
[ 9.96638293e-01],
[ 1.89550810e-03],
[ 1.00495808e+00],
[ 9.94525215e-01],
[ 9.90635852e-01],
[ 1.00173863e+00],
[ 1.00222370e+00],
[ 9.87304837e-01],
[ 9.96652895e-01],
[ 9.95185786e-01],
[ 1.00185924e+00],
[ 1.00648075e+00],
[ 9.98709680e-01],
[ 1.00203472e+00],
[ 9.95678167e-01],
[-1.62202386e-03],
[ 1.00219721e+00],
[ 9.97678835e-01],
[ 1.86113755e-02],
[ 9.92752805e-01],
[ 5.49299405e-03],
[-6.31492609e-03],
[ 7.84254043e-03],
[ 1.00813674e+00],
```

```
[ 1.00423985e+00],
[ 6.95439422e-07],
[ 1.00422688e+00],
[ 9.98657160e-01],
[ 1.00278431e+00],
[ 9.91619002e-01],
[ 1.00185714e+00],
[ 1.00363886e+00],
[ 1.00369892e+00],
[-2.14275995e-03],
[-1.78563235e-04],
[ 3.52771759e-03],
[ 9.97686910e-01],
[ 9.94024392e-01],
[ 3.62472408e-03],
[ 1.00261255e+00],
[ 9.95099944e-01],
[ 1.00078607e+00],
[ 9.91522144e-01],
[ 1.00409496e+00],
[ 9.94166662e-01],
[ 9.97901047e-01],
[ 2.78490838e-04],
[ 1.00114746e+00],
[ 1.00073153e+00],
[ 9.92437595e-01],
[ 9.93101935e-01],
[ 1.00193162e+00],
[ 9.99318451e-01],
[ 4.18560658e-03],
[ 9.96074881e-01],
[ 1.40008239e-03],
[ 1.10188006e-02],
[ 1.00216090e+00],
[ 1.00092964e+00],
[ 1.00487276e+00],
[ 1.00085412e+00],
[ 9.94604256e-01],
[ 1.00496461e+00],
[ 1.01009237e+00],
[ 9.85850781e-01],
[ 1.00128529e+00],
[ 9.95007574e-01],
[ 7.42429447e-03],
[ 6.16705222e-03],
[ 9.99207516e-01]])
```

What is the mean of the expected target value in test set ?

In [59]:
```
y_test.describe()
```

Out[59]:

	isAlive
count	502.000000
mean	0.749004
std	0.434019
min	0.000000
25%	0.250000
50%	1.000000
75%	1.000000
max	1.000000

Evaluating Linear Regression Accuracy using Root Mean Square Error

In [60]:
```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
```

In [61]:
```
print(RMSE)
```

```
0.00564167491606997
```

Decision Tree Regressor: Fitting a new regression model to the training set

In [62]:
```
regressor = DecisionTreeRegressor(max_depth=20)
regressor.fit(X_train, y_train)
```

Out[62]:
```
DecisionTreeRegressor(criterion='mse', max_depth=20, 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=False, random_state=None, splitter='best')
```

Performing Prediction using Decision Tree Regressor

In [63]:
```
y_prediction = regressor.predict(X_test)
y_prediction
```

Out[63]:
```
array([1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1.,
       0., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0.,
       1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 0., 1., 1., 1., 1.,
       1., 1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
       1., 1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
       0., 1., 1., 0., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1.,
       0., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1.,
       1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1.,
       1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
       1., 0., 1., 1., 0., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 0.,
       1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 0., 1., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
       1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 0., 1.])
```

For comparision: What is the mean of the expected target value in test set ?

In [64]:
```
y_test.describe()
```

Out[64]:

	isAlive
count	502.000000
mean	0.749004
std	0.434019
min	0.000000
25%	0.250000
50%	1.000000
75%	1.000000
max	1.000000

In [65]:
```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
```

In [66]:
```
print(RMSE)
```

```
0.0
```

Findind results without using actual column because of very high correlation

In [67]:
```
features=['alive', 'plod', 'male', 'book1', 'book2',
       'book3', 'book4', 'book5', 'isMarried', 'isNoble', 'numDeadRelations',
       'boolDeadRelations', 'isPopular', 'popularity']
```

In [68]:
```
X = data[features]
```

In [69]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, rand
om_state=324)
```

Linear Regression: Fitting a model to the training set

In [70]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[70]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [71]:
```
y_prediction = regressor.predict(X_test)
y_prediction
```

Out[71]:
```
array([[0.79681997],
       [0.73313088],
       [0.40850793],
       [0.65435515],
       [0.82347647],
       [0.64187971],
       [0.91163788],
       [0.77281962],
       [0.76175844],
       [0.86255235],
       [1.03991811],
       [0.43177695],
       [0.80686609],
       [0.95527705],
       [0.88567531],
       [1.02703361],
       [1.10434824],
       [0.70360069],
       [0.89891535],
       [0.70688037],
       [1.02786259],
       [0.86799241],
       [0.90331952],
       [0.69004702],
       [0.81778818],
       [0.87998891],
       [0.96754657],
       [0.94172819],
       [0.42426465],
       [0.90774862],
       [0.22569485],
       [0.77729583],
       [1.10022028],
       [0.36762239],
       [0.91367964],
       [1.03543411],
       [0.6958674 ],
       [0.68238385],
       [0.60269022],
       [0.43876829],
       [0.30070861],
       [0.63050854],
       [0.68580852],
       [1.0506437 ],
       [0.34462934],
       [0.76275196],
       [0.76003719],
       [0.77715137],
       [0.57273955],
       [0.99154661],
       [0.85814724],
       [0.95189123],
       [0.4972499 ],
       [1.0309097 ],
       [0.65523442],
       [0.8482008 ],
       [0.44063666],
```

```
[0.94761172],
[0.60695075],
[0.89719985],
[0.76075156],
[0.67378921],
[0.75298587],
[0.86648579],
[0.90299179],
[0.80045529],
[1.00257286],
[0.63468489],
[0.51874302],
[1.06096586],
[0.81863822],
[1.01710896],
[0.67881697],
[0.67392499],
[0.70227738],
[0.4741889 ],
[0.89665008],
[0.54715547],
[0.66454359],
[0.66668077],
[0.43563571],
[0.44059574],
[0.69166506],
[0.9145701 ],
[0.44433944],
[0.93702399],
[0.73926273],
[0.58439675],
[1.01671516],
[0.92575883],
[0.88344712],
[0.92451203],
[1.031838  ],
[0.60641848],
[0.34858736],
[0.77591833],
[0.95172876],
[0.62517654],
[0.95289637],
[0.86056605],
[0.64862311],
[0.90089496],
[0.8229312 ],
[0.97922201],
[0.5892953 ],
[0.99926003],
[1.04321038],
[0.75247008],
[0.85906298],
[0.75180962],
[0.37988205],
[0.97538075],
[0.89889551],
[0.63926603],
```

```
[1.04993244],
[0.75271238],
[0.94999789],
[0.75275456],
[0.78774596],
[1.00880788],
[0.88882923],
[0.68211781],
[0.95507327],
[0.77475072],
[0.78249424],
[0.92411481],
[0.84477103],
[0.80758913],
[0.97083885],
[0.37995159],
[0.80765711],
[0.76376704],
[0.81007556],
[0.33790757],
[0.84270022],
[0.68540096],
[1.01613135],
[0.73038592],
[0.77091189],
[0.28888098],
[0.42498869],
[0.86766452],
[0.77694109],
[0.56929982],
[0.69185737],
[0.80434337],
[0.99826945],
[0.65737964],
[0.79799674],
[0.78398475],
[1.00428996],
[0.34641853],
[0.39101199],
[0.92971114],
[0.27467692],
[0.78876369],
[0.60256944],
[0.2472982 ],
[0.38529152],
[0.66372846],
[0.79654347],
[0.63952451],
[0.89309518],
[0.7837375 ],
[0.93266689],
[0.56165529],
[0.73349548],
[0.63567597],
[0.88650787],
[0.7809701 ],
[0.76998259],
```

```
[0.38693183],
[0.65233329],
[0.64335733],
[0.39640267],
[1.01040067],
[0.74145793],
[0.77556242],
[0.5735605 ],
[0.68610907],
[0.75580919],
[0.76351075],
[0.68042354],
[0.87453889],
[0.86600289],
[0.89807623],
[0.50059412],
[0.83351177],
[0.77281962],
[0.80207767],
[0.4095614 ],
[0.94954223],
[0.78437168],
[0.87279763],
[0.87722301],
[1.00505031],
[0.89027969],
[1.09941892],
[0.66168946],
[0.77594038],
[0.7246894 ],
[0.90667145],
[1.05458165],
[0.75599981],
[0.78454622],
[0.54698622],
[0.79426807],
[0.81012728],
[0.67510042],
[0.28939183],
[0.80221726],
[0.41789099],
[0.68813699],
[0.35249965],
[0.51886284],
[0.67519728],
[1.03262559],
[1.00653835],
[0.7690292 ],
[0.54758826],
[0.84591578],
[0.77145844],
[0.91144787],
[0.72344907],
[0.66125917],
[1.04269954],
[0.86598912],
[0.73993556],
```

```
       [0.379319  ],
       [0.92115818],
       [0.48754933],
       [0.7368208 ],
       [0.78204593],
       [1.01186147],
       [0.87939133],
       [0.89930307],
       [0.73143326],
       [0.44960446],
       [0.81186719],
       [1.04617073],
       [1.01186147],
       [0.42856156],
       [0.6664507 ],
       [0.37939532],
       [0.30188999],
       [0.74917165],
       [1.03668512],
       [0.95269259],
       [0.47845476],
       [0.83537657],
       [0.77237418],
       [0.34925978],
       [0.87945379],
       [0.46370454],
       [0.17172855],
       [0.5443156 ],
       [0.65043999],
       [0.77216913],
       [0.69936995],
       [0.70021481],
       [1.01225606],
       [0.95719732],
       [0.18225185],
       [0.83213936],
       [0.95209501],
       [0.93298906],
       [0.66876941],
       [1.04309601],
       [0.54549164],
       [0.60329892],
       [1.0341223 ],
       [0.51718859],
       [0.61664251],
       [0.80780836],
       [0.68485226],
       [0.78391027],
       [0.79883793],
       [0.74026561],
       [0.8733116 ],
       [1.01139204],
       [0.41529942],
       [0.63059926],
       [0.70882161],
       [0.7788963 ],
       [0.41480865],
```

```
       [0.85688696],
       [0.46423682],
       [0.84224397],
       [0.91256461],
       [0.8439363 ],
       [0.6339678 ],
       [0.14899278],
       [0.8394721 ],
       [0.54958169],
       [1.00486   ],
       [0.8149066 ],
       [0.57845076],
       [1.1013879 ],
       [0.77769396],
       [1.1120479 ],
       [0.52887298],
       [0.39018233],
       [0.61950066],
       [0.6427873 ],
       [0.49617583],
       [0.49377929],
       [0.60255942],
       [0.81288127],
       [0.98468036],
       [0.42971779],
       [0.53252928],
       [0.79505437],
       [0.37919423],
       [0.66624691],
       [0.16978544],
       [0.36803789],
       [1.024989  ],
       [0.91906462],
       [0.98532253],
       [0.61897371],
       [0.87001421],
       [0.89977315],
       [0.77356106],
       [0.6936592 ],
       [0.76104737],
       [0.53002459],
       [0.79354804],
       [0.37420323],
       [0.87983724],
       [0.80077128],
       [0.76705106],
       [0.5319145 ],
       [0.44845748],
       [0.54048122],
       [0.62697146],
       [0.69590006],
       [0.4321537 ],
       [0.36711814],
       [0.39336748],
       [0.6852487 ],
       [0.70702882],
       [0.66704963],
```

```
[0.50788839],
[0.57807527],
[0.21676417],
[0.85637941],
[1.01729554],
[0.32974502],
[0.44878557],
[0.76749001],
[0.36857216],
[0.58333801],
[1.01295345],
[0.972034  ],
[0.30773732],
[0.54319301],
[0.97217533],
[0.95507327],
[1.01296722],
[0.74791669],
[0.39514279],
[1.04972866],
[0.94449853],
[0.69967207],
[1.04283036],
[0.86573689],
[0.82173882],
[0.60743739],
[0.80509259],
[1.03435688],
[0.42692334],
[0.94234453],
[0.97738483],
[0.64479197],
[0.49138644],
[0.67231164],
[0.51907735],
[0.48557949],
[0.75094308],
[0.67162952],
[0.43236696],
[0.69607284],
[0.49866261],
[0.72697721],
[0.72815859],
[0.49376761],
[0.82051266],
[0.68835746],
[0.70259032],
[0.84223392],
[0.60969567],
[0.82422275],
[0.4277075 ],
[0.69289771],
[0.76893392],
[0.84171801],
[0.66732302],
[0.34932705],
[1.03970399],
```

```
       [0.95349395],
       [0.87907375],
       [0.84278661],
       [0.60256944],
       [0.91785598],
       [0.6817019 ],
       [0.80942317],
       [1.04631588],
       [1.01336102],
       [0.69308303],
       [0.93011812],
       [0.81389378],
       [0.62985262],
       [1.01091699],
       [0.57623884],
       [0.66078028],
       [0.94460344],
       [0.72698714],
       [0.94428097],
       [0.71466178],
       [0.85679878],
       [0.29612498],
       [0.56746356],
       [0.72018855],
       [0.49301554],
       [0.9678785 ],
       [1.01610381],
       [0.37713994],
       [0.39295342],
       [0.72077367],
       [1.10822433],
       [0.41420572],
       [0.58533983],
       [0.94196266],
       [0.89362747],
       [0.3889428 ],
       [0.71775146],
       [0.5803399 ],
       [1.09622047],
       [0.82469679],
       [0.64395444],
       [0.71511694],
       [0.76021095],
       [0.84332496],
       [0.9356277 ],
       [1.11000329],
       [1.00968871],
       [0.89260855],
       [0.53963709],
       [0.92103053],
       [0.92691298],
       [0.57394157],
       [0.58331478],
       [0.40035385],
       [0.24488819],
       [0.37298169],
       [0.68590005],
```

```
       [0.53021775],
       [0.43469971],
       [0.38861584],
       [0.86220332],
       [0.65368512],
       [0.70626825],
       [1.02376246],
       [0.85423778],
       [0.46154487],
       [1.03377986],
       [0.55005306],
       [0.3853294 ],
       [0.65159755],
       [0.94448476],
       [0.32016437],
       [0.8736461 ],
       [0.9979792 ],
       [1.01276344],
       [0.68314872],
       [0.70459322],
       [0.86706914],
       [0.54805703],
       [0.56187361],
       [0.44297189],
       [0.93530689],
       [0.66235322],
       [0.65786284],
       [0.62052433],
       [0.83580153],
       [0.86331744],
       [0.87905328],
       [0.59459292],
       [0.9851014 ],
       [0.87569523],
       [0.9362364 ],
       [0.53911515],
       [1.02930698],
       [0.57146338],
       [0.68751695],
       [0.85867241],
       [0.77636378],
       [0.92814294],
       [0.86635499],
       [0.32156608],
       [0.41990045],
       [0.78706488]])
```

In [72]:
```
y_test.describe()
```

Out[72]:

	isAlive
count	502.000000
mean	0.749004
std	0.434019
min	0.000000
25%	0.250000
50%	1.000000
75%	1.000000
max	1.000000

In [73]:
```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
```

In [74]:
```
print(RMSE)
```

```
0.40168202561094757
```

Decision Tree Regressor: Fitting a new regression model to the training set

In [75]:
```
regressor = DecisionTreeRegressor(max_depth=100)
regressor.fit(X_train, y_train)
```

Out[75]:
```
DecisionTreeRegressor(criterion='mse', max_depth=100, 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=False, random_state=None, splitter='best')
```

In [76]:
```
y_test.describe()
```

Out[76]:

	isAlive
count	502.000000
mean	0.749004
std	0.434019
min	0.000000
25%	0.250000
50%	1.000000
75%	1.000000
max	1.000000

In [77]:
```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
```

In [78]:
```
print(RMSE)
```

```
0.40168202561094757
```