Random Forest, Xgboost and Feature Engineering on Titanic Dataset

Hongye Xu
Rutgers University
New Brunswick, NJ 08901
hongye.xu@rutgers.edu

1 Introduction

The sinking of the Titanic is one of the most infamous shipwrecks in history. We admit there was some element of luck involved in surviving while it seems some groups of people were more likely to survive than others: women, children and upper class passengers. My project aims to build a predictive model that answers the question: "what sorts of people were more likely to survive?"

2 Dataset

Data source: Titanic: Machine Learning from Disaster[1]

2.1 Data Dictionary

- `PassengerId` : the unique id of the row and it doesn't have any effect on `Survived`.
- `Survived` : binary (**0** or **1**);
  - **1** = **Survived**
  - **0** = **Not Survived**
- `Pclass` (Passenger Class) : the socio-economic status of the passenger. It is a categorical ordinal feature which has **3** unique values (**1**, **2** or **3**);
  - **1** = **Upper Class**
  - **2** = **Middle Class**
  - **3** = **Lower Class**
- `Name`, `Sex` and `Age` features are self-explanatory.
- `SibSp` : the total number of the passengers' siblings and spouse.
- `Parch` : the total number of the passengers' parents and children.
- `Ticket` : the ticket number of the passenger.
- `Fare` : the passenger fare.
- `Cabin` : the cabin number of the passenger.
- `Embarked` is port of embarkation. It is a categorical feature and it has **3** unique values (**C**, **Q** or **S**);
  - **C = Cherbourg**
  - **Q = Queenstown**
  - **S = Southampton**

2.2 Taking a look at the Dataset

There are 418 samples in test_data and 891 samples in the train_data. I merge these two data sets for missing value filling and feature engineering. Let's take a look at the merged dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
```

[1] https://www.kaggle.com/c/titanic/data

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

```
Data columns (total 12 columns):
Age             1046 non-null float64
Cabin           295 non-null object
Embarked        1307 non-null object
Fare            1308 non-null float64
Name            1309 non-null object
Parch           1309 non-null int64
PassengerId     1309 non-null int64
Pclass          1309 non-null int64
Sex             1309 non-null object
SibSp           1309 non-null int64
Survived        891 non-null float64
Ticket          1309 non-null object
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
0	22.0	NaN	S	7.2500	Braund, Mr. Owen Harris	0	1	3	male	1	0.0	A/5 21171
1	38.0	C85	C	71.2833	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	2	1	female	1	1.0	PC 17599
2	26.0	NaN	S	7.9250	Heikkinen, Miss. Laina	0	3	3	female	0	1.0	STON/O2. 3101282
3	35.0	C123	S	53.1000	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	4	1	female	1	1.0	113803
4	35.0	NaN	S	8.0500	Allen, Mr. William Henry	0	5	3	male	0	0.0	373450

Figure 1: merged dataset

3 Missing Value Processing

```
Age             263
Cabin          1014
Embarked          2
Fare              1
Name              0
Parch             0
PassengerId       0
Pclass            0
Sex               0
SibSp             0
Survived        418
Ticket            0
dtype: int64
```

As shown above, there exists missing values in 5 features, We need to fill the missing values in different ways.

3.1 Age

The age feature has 263 null values. Referring to Fill missing values using Random Forest[2], I use a random forest regression model to simulate the value. The features we use here are sex, pclass, Parch, SibSp.

[7]:
```
from sklearn.ensemble import RandomForestRegressor

age_df = all_data[['Age', 'Pclass','Sex','Parch','SibSp']]
age_df=pd.get_dummies(age_df)
known_age = age_df[age_df.Age.notnull()].as_matrix()
unknown_age = age_df[age_df.Age.isnull()].as_matrix()
y = known_age[:, 0]
X = known_age[:, 1:]
```

[2]https://www.mikulskibartosz.name/fill-missing-values-using-random-forest

```
rfr = RandomForestRegressor(random_state=0, n_estimators=100, n_jobs=-1)
rfr.fit(X, y)
predictedAges = rfr.predict(unknown_age[:, 1::])
all_data.loc[(all_data.Age.isnull()), 'Age'] = predictedAges
```

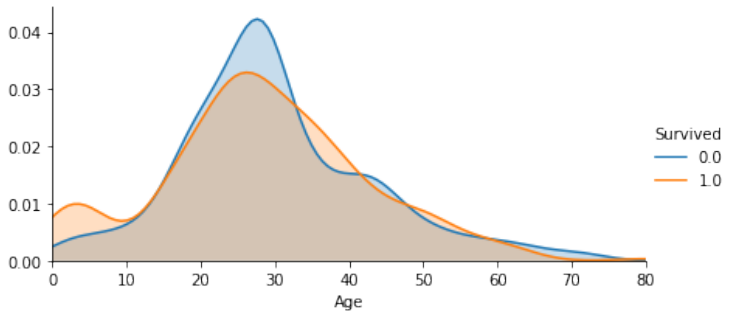


Figure 2: Distribution of age after filling in missing values.

3.2 Fare

I fill the missing value in Fare with the median Fare of 3rd class alone passenger.

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
1043	60.5	NaN	S	NaN	Storey, Mr. Thomas	0	1044	3	male	0	NaN	3701

Figure 3: Missing value in fare column.

3.3 Embarked

When I googled **Stone, Mrs. George Nelson (Martha Evelyn)**, I learned that **Mrs Stone** boarded the Titanic in **Southampton** on 10 April 1912 and was travelling in first class with her maid **Amelie Icard** in this page Martha Evelyn Stone: Titanic Survivor[3]. So I can simply use 'S' to fill in missing values.

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
61	38.0	B28	NaN	80.0	Icard, Miss. Amelie	0	62	1	female	0	1.0	113572
829	62.0	B28	NaN	80.0	Stone, Mrs. George Nelson (Martha Evelyn)	0	830	1	female	0	1.0	113572

Figure 4: Missing value in embarked column.

4 Feature Engineering

4.1 Title Extraction

Here I refer to a very interesting kernel for title extraction:Titanic [EDA] + Model Pipeline + Keras NN[4]. Take a look at the Name column.

```
0                               Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
Name: Name, dtype: object
```

Then I extract the prefix of all passengers and put different prefix into different categories.

[3]https://www.encyclopedia-titanica.org/titanic-survivor/martha-evelyn-stone.html
[4]https://www.kaggle.com/kabure/titanic-eda-model-pipeline-keras-nn#5.-Preprocessing-:

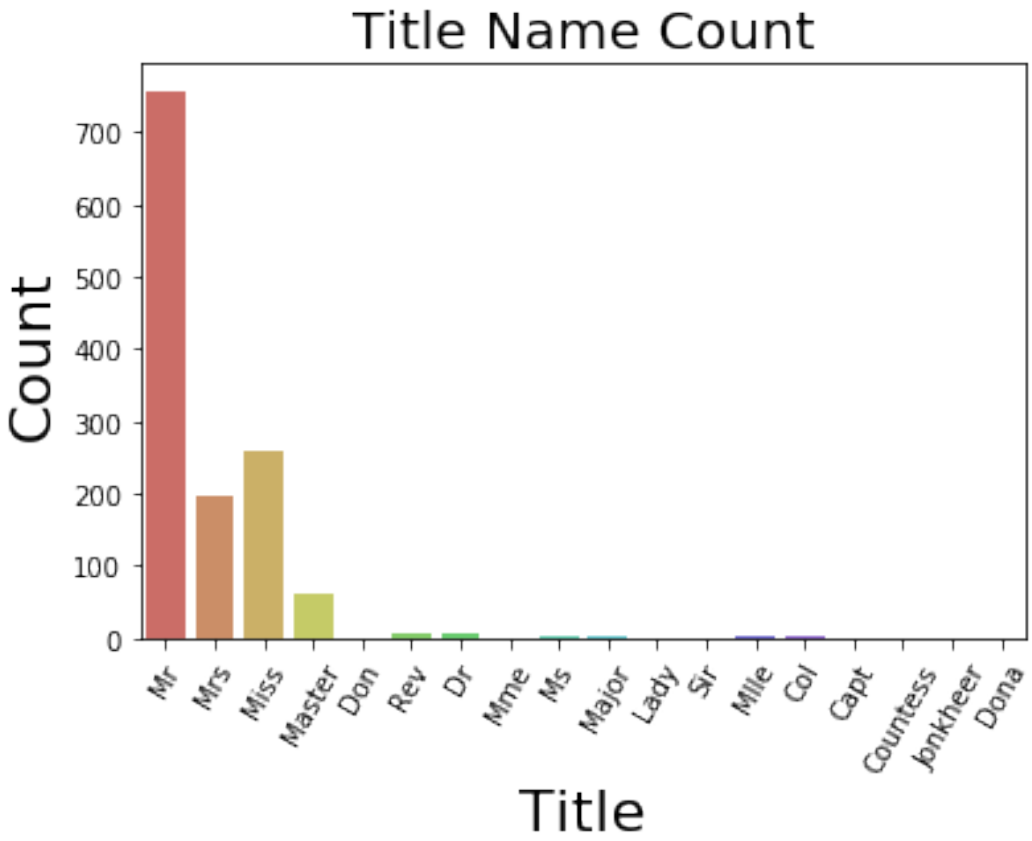


Figure 5: Prefix count

Prefix	Capt	Col	Major	Jonkheer	Don	Sir	Dr
Title	Officer	Officer	Officer	Master	Royalty	Royalty	Officer
Prefix	Rev	the Countess	Mme	Mlle	Ms	Mr	Mrs
Title	Officer	Royalty	Mrs	Miss	Mrs	Mr	Mrs

Table 1: Title Dictionary

```
[17]: Mr          757
      Miss        262
      Mrs         200
      Master       62
      Officer      23
      Royalty       4
      Name: Title, dtype: int64
```

4.2 FamilyName

For the processing of this feature, I refer to Titanic using Name only [0.81818][5]. The author Chris Deotte builds a simple model(Figure 6) only with the name and sex column and scores 82% accuracy.

He found that the male older than 16 are dead and the female except every one under the familyname dead are survived. Thus I first create a new feature 'FamilyGroup' as frequency

[5]https://www.kaggle.com/cdeotte/titanic-using-name-only-0-81818

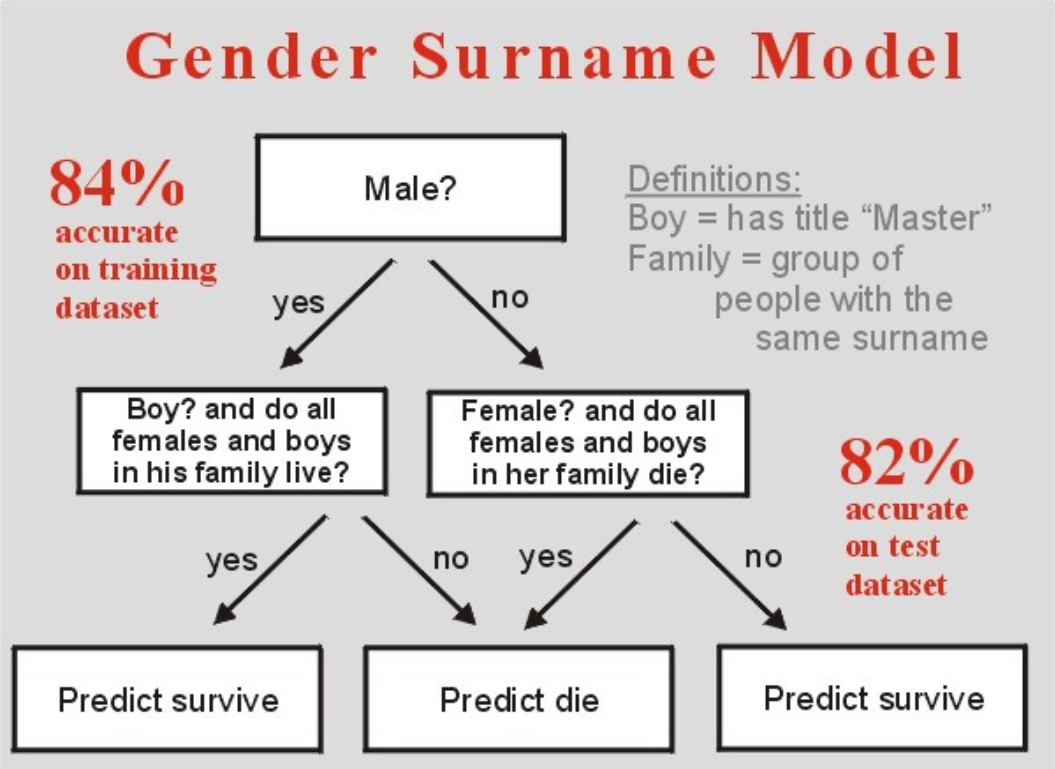


Figure 6: Name Sex model

of family names(Figure 7). This feature helps to find out those who are not single. Then I put the passengers whose family groups are not less than 2 into 2 groups: FemaleChild and MaleAdult. Finally I make the exception list of MaleAdult group and FemaleChild group. Then I change the features of test family names in the list into the dead features or survived features according to the list.(Since that the passengers in the list tend to obey the average fate of their family name).

The Dead List of Female_Child_Group(16 years old or younger, female) :

```
{'Bourke', 'Olsson', 'Danbom', 'Skoog', 'Van Impe', 'Lahtinen', 'Strom',
'Panula', 'Sage', 'Turpin', 'Rice', 'Vander Planke', 'Oreskovic', 'Caram',
'Zabour', 'Canavan', 'Rosblom', 'Lobb', 'Palsson', 'Barbara',
'Arnold-Franchi', 'Robins', 'Ford', 'Johnston', 'Attalah', 'Boulos',
'Cacic',  'Goodwin', 'Elias', 'Ilmakangas', 'Jussila', 'Lefebre'}
```

The Survived List of Female_Child_Group(Over 16 years old, male) :

```
{'Moubarek', 'Beckwith', 'Daly', 'Nakid', 'Harder', 'Jonsson', 'Beane',
'Goldenberg', 'Cardeza', 'Chambers', 'Taylor', 'Bishop', 'Greenfield',
'Kimball', 'McCoy', 'Frolicher-Stehli', 'Frauenthal', 'Duff Gordon',
'Dick','Jussila', 'Bradley'}
```

4.3 Family Size

The kernel Titanic [EDA] + Model Pipeline + Keras NN[6] also provide a good idea about the familysize. The familysize is defined as the sum of the total number of the passengers' siblings and spouse and the total number of the passengers' parents and children plus one(the passenger himself/herself).

We can see that in Figure 8a and 8b the families with size 2 to 4 have relatively higher survival rate, so we can label the family size with 3 different type.

[6]https://www.kaggle.com/kabure/titanic-eda-model-pipeline-keras-nn#5.-Preprocessing-:

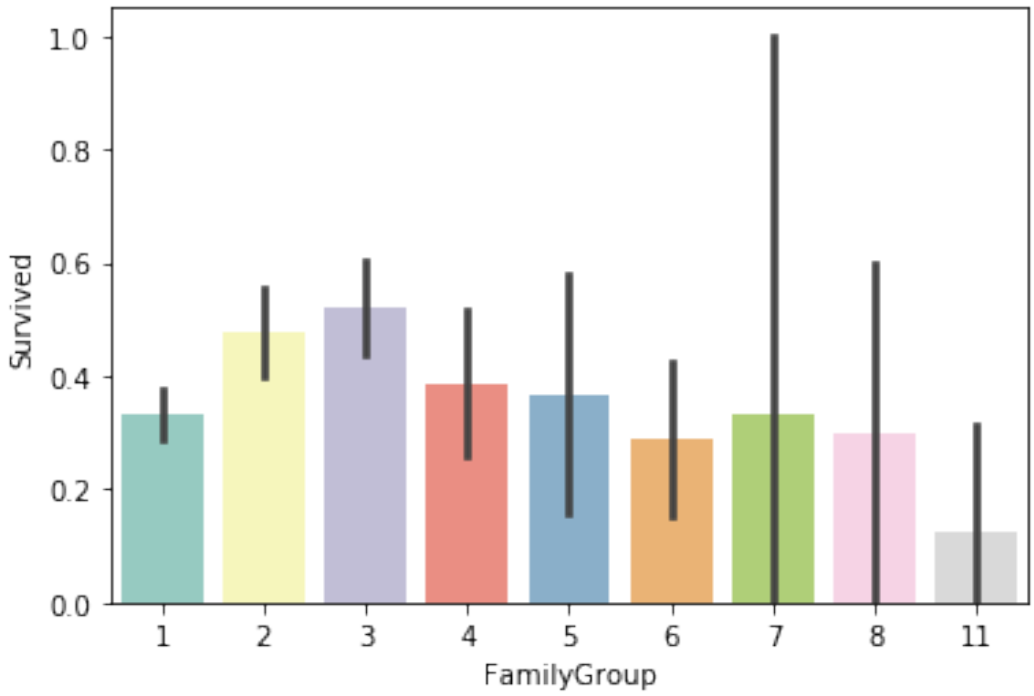
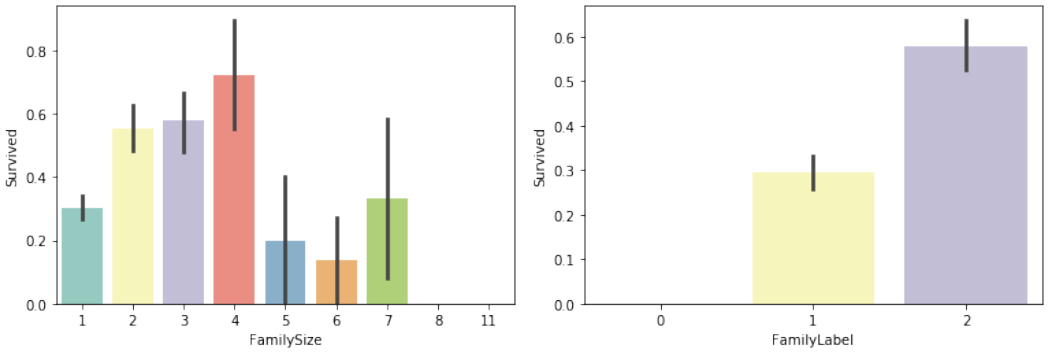


Figure 7: Survival rate by FamilyGroup



(a) Survival rate by FamilySize

(b) Survival rate by FamilyLabel

Figure 8: FamilySize and FamilyLabel

4.4 Cabin and Deck

There are many null values in 'Cabin' features. For the better predictions, it is believed to delete the feature of 'Cabin'. But here we need this feature because when the ship sinks, certain parts of the ship have different probability drown in water. So we deal to make a new feature to substitute the feature. Here I refer to another kernel's engineering on `cabin`(Titanic: Tutorial, Encoding, Feature Eng, 81.8%)[7] and simplified the code.I fill the missing value with string "Unknown" and than extract the first letter of this feature and put it in a new column named "Deck".

[7]https://www.kaggle.com/volhaleusha/titanic-tutorial-encoding-feature-eng-81-8

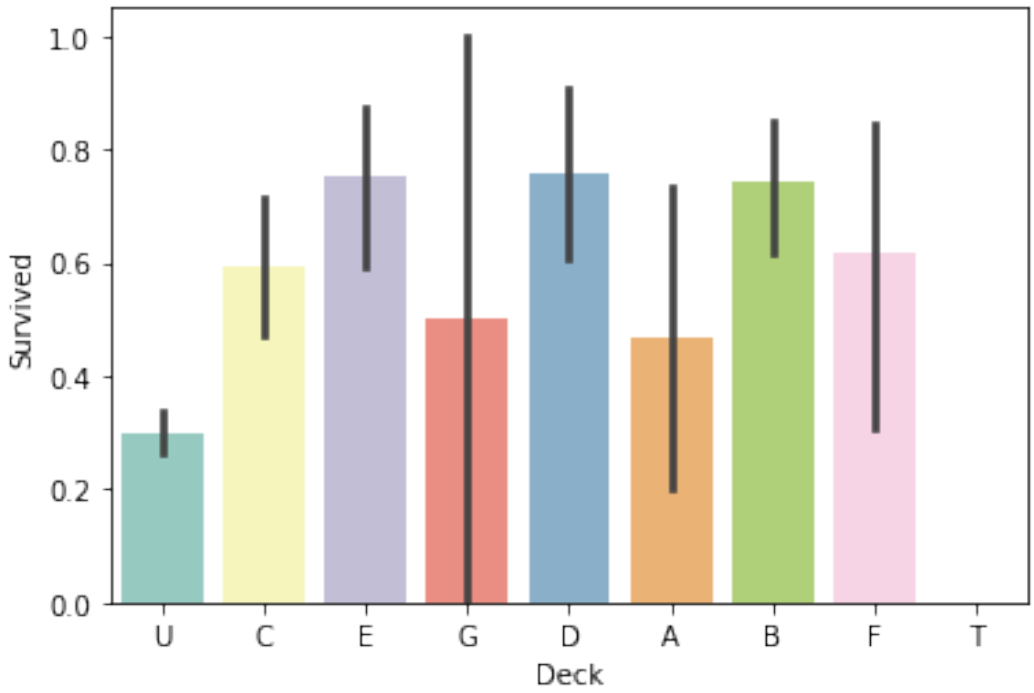
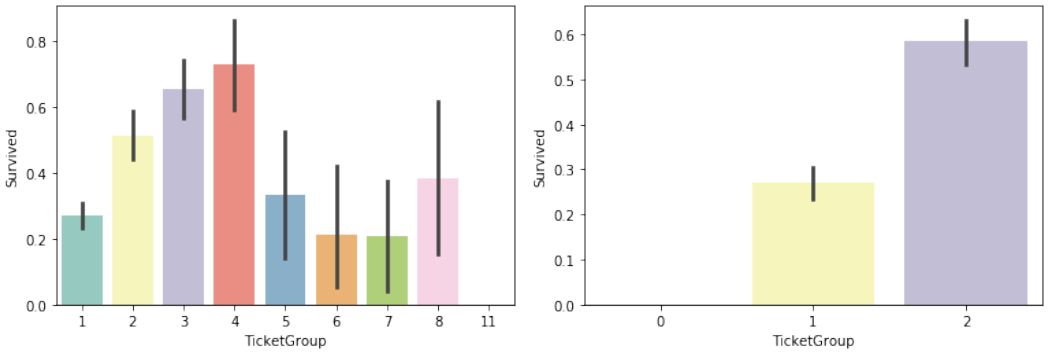


Figure 9: Survival rate by deck

4.5 Ticket Group

I create a new feature 'TicketGroup' as frequency of tickets and finally label this column into 3 types according to different survival performance. Survival rate by ticketgroup is shown in Figure 10a and 10b.



(a) Survival rate by Ticketgroup

(b) Survival rate by Ticketgroup(Labeled)

Figure 10: Ticketgroup

5 Modeling

I train and test two different tree models on the dataset with 10-fold cross validation to predict the survival of the passengers.

Random Forest The random forests is a classification algorithm consisting of many decisions trees[1]. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more

	Validation Score	Standard Deviation	Leaderboard Score
Xgboost	0.8429	0.0366	0.7751
Random Forests	0.8407	0.0353	0.8421

Table 2: Results of different models

accurate than that of any individual tree. My final model consists of 26 classification trees with a depth of 6.

Xgboost The Xgboost is a scalable and accurate implementation of gradient boosting machines and it has been proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. It is a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning and achieve state-of-the-art results on many machine learning challenges[2].

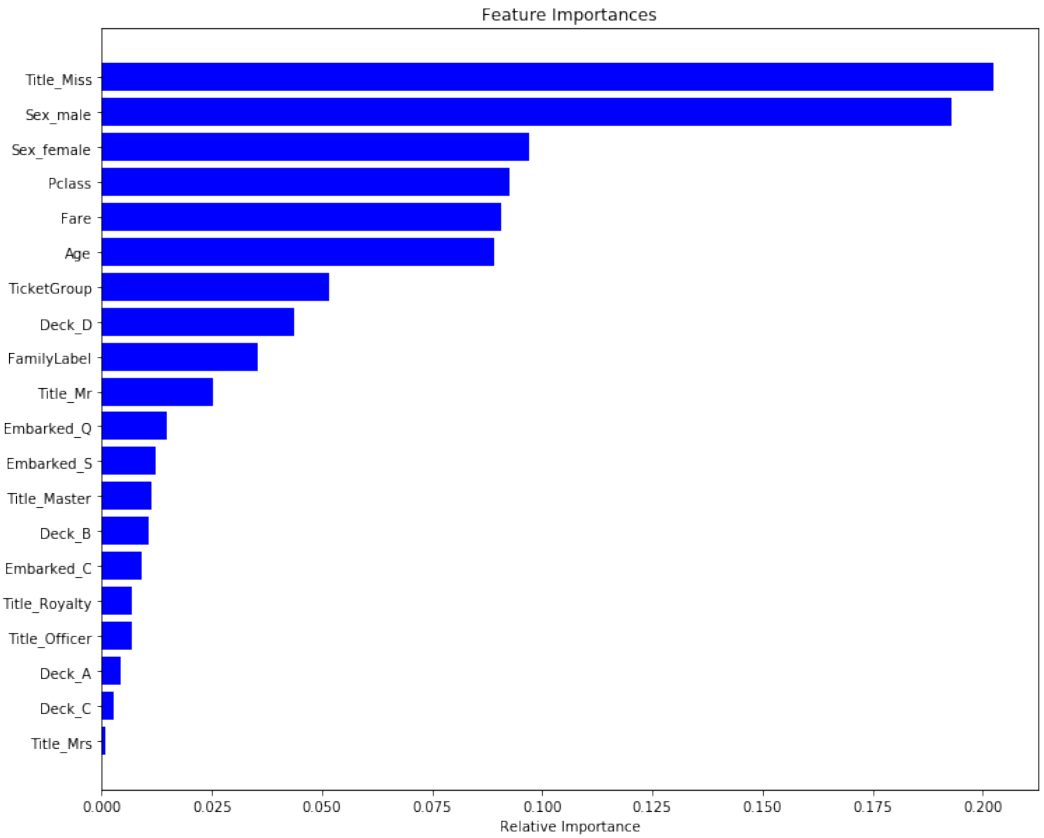


Figure 11: Partial Dependent Plot for Random Forests

6 Conclusion

According to the partial dependent plot, we can see that both models treat the Title_Miss, Sex_female, Pclass as important features. This validates our initial hypothesis that some groups of people were more likely to survive than others: women, children and upper class passengers. One thing that is worth mentioning is that the Xgboost model does not treat Sex_male as important and I believe this might be the reason that the Xgboost does not achieve better accuracy than the Random Forests.

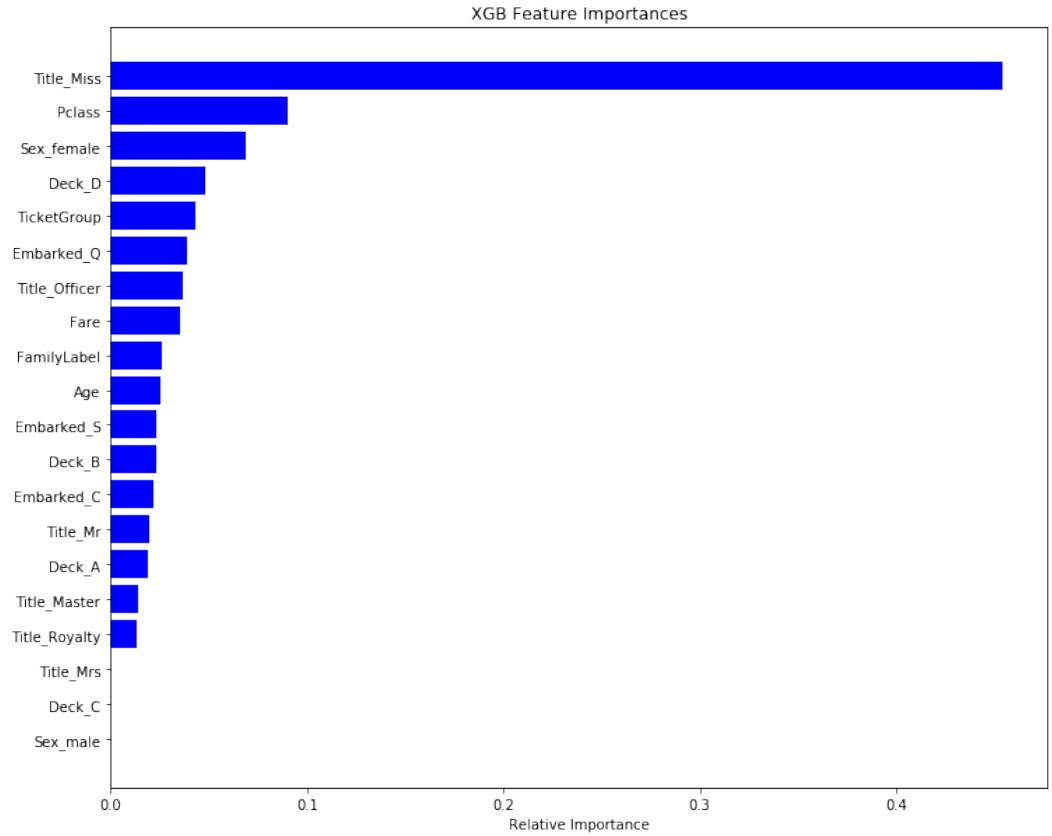


Figure 12: Partial Dependent Plot of Xgboost

References

[1] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. 2016.