2/2/2018

Titanic: Machine Learning from Disaster | Kaggle

Eklavya Saxena
NUID: 1850025
CSYE 7245
BIG-DATA SYSTEMS AND INTELLIGENCE ANALYTICS

Abstract

As per the exploration:

- Number of Observations: **891**
- Number of Features (or Variables): **11 (excl. Target Variable/Label)**
- Target Variable: **“survival” (0 = No, 1 = Yes)**
- Different Data Types of Features: **2 (Categorical & Numerical, excl. Textual)**
- Age has (891 – 714) **177 missing values**
- Only **38.38%** of the passengers survived

Identification of the problem:

- The problem statement requires to predict if a passenger survived the sinking of the Titanic or not. For each PassengerId in the test set, predict a 0 or 1 value for the Survived variable.
- One sample belongs to one class only and there are only two classes (namely 0 or 1).
- Therefore, it is a **binary classification problem with single column**.

Identification of different variables in the data:

- Age, Sibsp, Parch - **Numerical** (excl. Fare, out of context)
- Pclass, Sex, EmbarkedPort - **Categorical**
- Survived - **Target**
- Name, Ticket, Cabin - Textual (out of context)

Machine Learning Algorithm used:

- Decision Tree
- Random Forest
- Logisitic Regression

Introduction

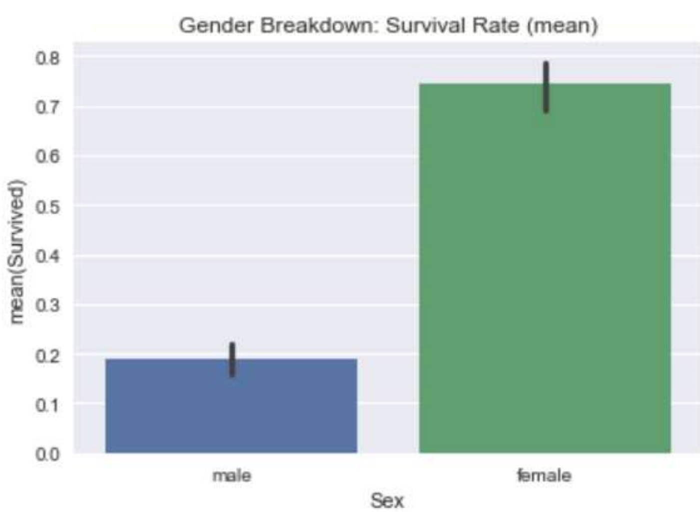
Around, 37% of the data in “Age” column is missing. The estimation of the “Age” is done using the **salutation** in “Name”, “Sex”, and “Pclass”. A new feature is introduced “family_size” which is the total of “SibSp” + “Parch” + 1 (the observation itself). It has been assumed that larger families need more time to get together on a sinking ship, and hence have lower probability of surviving.

Code Documentation

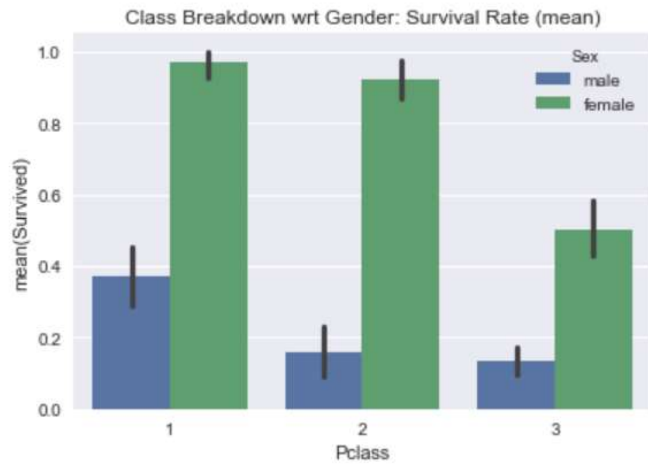
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

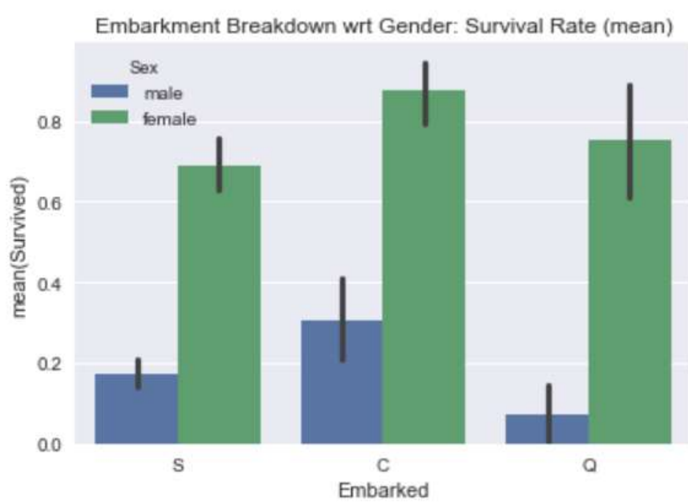
```
sns.barplot(x='Sex', y='Survived', data=train);
plt.ylabel('mean(Survived)');
plt.title("Gender Breakdown: Survival Rate (mean)");
```



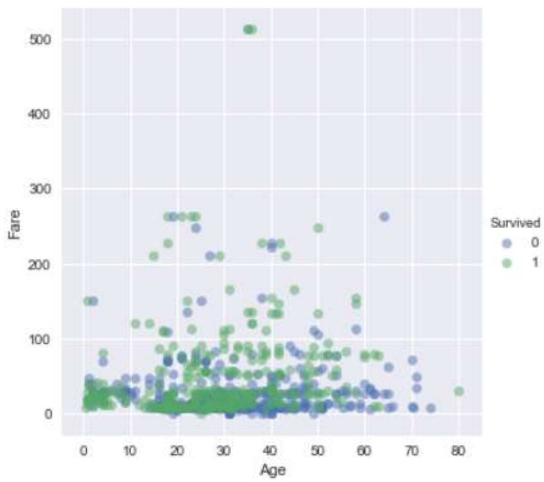
```python
# Survival Rate grouped by Pclass and Sex
sns.barplot(x='Pclass', y='Survived', hue='Sex', data=train);
plt.ylabel('mean(Survived)');
plt.title("Class Breakdown wrt Gender: Survival Rate (mean)");
```




```python
# Survival Rate grouped by Embarked and Sex
sns.barplot(x='Embarked', y='Survived', hue='Sex', data=train);
plt.ylabel('mean(Survived)');
plt.title("Embarkment Breakdown wrt Gender: Survival Rate (mean)");
```



```python
sns.lmplot(x='Age', y='Fare', hue='Survived', data=train, fit_reg=False, scatter_kws={'alpha': 0.5});
```



It can inferred that, those who survived either paid quite a bit or they were young.

```python
plt.figure(figsize=(12,10));
sns.heatmap(train.corr(), annot=True, fmt=".2f");
plt.show();
```

- Corelation -0.34: Survived is inversely propotional to the Pclass:
  - We have already examined the Pclass dependency upon Survived
  - As Pclass and Fare are closely related, we will examine the numeric variable Fare here
- Corelation -0.42: Pclass is inversely propotional to the Age:
  - From above 2 inferences, Age is directly propotional to Survived
- Corelation -0.31: Age is inversely propotional to the SbSp:
  - This provides an insight, that younger passenger have more chances of having siblings/spouses
- Corelation +0.41: SbSp is directely propotional to the Parch:
  - Also, passengers with siblings/spouses are more likely to have parents/children

```
sns.pairplot(hue='Survived', data=train);
```

Results

Using the imputed dataset, Decision Tree, Random Forest, and Logistic Regression is used, which an accuracy of 76.88%, 79.8%, 80.02%.

```
# Comparision
models = []
models.append(('Decision Tree', DecisionTreeClassifier()))
models.append(('Random Forest', RandomForestClassifier()))
models.append(('Logistic Regression', LogisticRegression()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=5, random_state=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
Decision Tree: 0.768822 (0.013516)
Random Forest: 0.798010 (0.025343)
Logistic Regression: 0.802479 (0.025878)
```

References

https://www.datacamp.com/community/tutorials/kaggle-machine-learning-eda

https://campus.datacamp.com/courses/kaggle-python-tutorial-on-machine-learning/predicting-with-decision-trees?ex=1

https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/

https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/

https://www.analyticsvidhya.com/blog/2014/09/data-munging-python-using-pandas-baby-steps-python/