

ASSIGNMENT 1: Titanic - Machine Learning from Disaster

PROBLEM DEFINITION

The problem states to apply the basics of Machine Learning concepts and build a predictive model that can help understand whether titanic passengers' survival rate depends on other factors like age, fare, ticket class, etc. The train.csv file gives details of 891 passengers and whether they survived or not. With the help of train dataset, a model can be developed to predict passengers' survival outcome with details given in test.csv.

APPROACH

Understanding the problem and datasets

The train.csv file contains information like age, ticket class, family information, cabin and fare details of 891 different passengers in titanic. The survival column states whether the person was alive after the disaster or not. It is represented by the numbers 0 (did not survive) and 1 (survived).

The test.csv file contains similar information of other 418 passengers whose survival values we need to predict with the model trained by train.csv. This presents a classification machine learning problem as the data to be predicted has two categories: 0 or 1.

To get an overview of data, I used various functions to retrieve details like shape, data types, and other statistics like mean, median, etc.

For better observation, I plotted some graphs with matplotlib, to understand the range of values present in particular columns.

Using the corr() function, I checked for correlations between columns to understand how the data might get affected if I were to perform some data cleaning or transformation techniques.

With the help of pandas' `crosstab()` function, I observed that:

- Sex : Females had a higher survival rate than males.
- PClass : People travelling in 3rd class had the lowest survival rate.
- Embarked : Passengers who boarded from Southampton had highest percentage of death.
- Parch/SibSp : Passengers travelling individually/ without their families had better chance of survival.

- Data pre-processing

Any data mining or machine learning problem requires data pre-processing. This is because raw data may contain abnormalities like missing data, extreme values or text value in numerical data.

Using the `is_null()` function, we can check for missing values in columns. It was observed that cabin had the highest number of missing values followed by age and then embarked with only two missing values.

Taking care of missing values:

1. Cabin: The columns consisted of less than 50% of the data. Its correlation with survival is not very strong so rather than replacing a large number of values; I decided to delete the column.
2. Age: It is a continuous variable and had some data missing. It can be filled with mean or median. First, I replaced with the mean of age; it was observed that the mean of age grouped by ticket class i.e PClass gave more relatable values.
3. Embarked: As it is a categorical column and has only two missing values, I thought of replacing it with the most occurring value, which was 'S'.

I deleted a few columns like 'Name' and 'Ticket' because almost every value was unique and it won't have a significant effect on the training of model or prediction of outcome.

While looking at the 'Fare' variable's boxplot, the feature had a wide range of values. Some of them were very extreme, which may be because of errors at the time of data entry. So, I replaced the values above 300 with the median of fare column.

With the help of binning, I formed a new column of age group, dividing child, adult, and elderly to help in better organization. (And deleted age column)

The two columns – SibSp and Parch – gave information about siblings, spouse and parent, child, respectively. As both columns reflected on the person's family, I decided to merge it into one column stating the number of family members for generalization.

- Splitting dataset into features (sex, age group, family info, Pclass, Embarked, fare) and response (Survival).
- Data encoding:
The dataframe consists of many categorical features which can be easier for the machine to depict after encoding. So, columns like Embarked, Sex, etc. with categorical data can be encoded as numbers. For example, sex is now represented as male : 0 and female: 1, same for other columns.
- All the above steps were applied to test dataset as well. Errors or warnings may be raised if the train data and test data do not have a similar transformation. It may lead to misleading results sometimes.
- I have used StandardScaler() to normalize both train and test data, leading to better accuracy.

CHECK FOR BEST SUITABLE MODEL

I created objects of a few machine learning models importing from sklearn, and placed them in an array. With the help of 'for loop' and cross_val_score function, I checked for the highest accuracy of the model when training data is divided into cross-validation sets. I tried different values of a number of cross-validation sets and found 10 to get the best results.

SELECTION OF MODEL

The highest accuracy is given by Random Forest classifier. Random Forest classifier is an ensemble learning method that works with multiple decision trees and considers mode of the output generated by them. Advantages of this model include high accuracy, less training time, low bias and high variance.

I used various parameters like `n_estimators`(no. of trees), `max_depth`(maximum depth of the tree), `criterion` (for measuring the quality of split), and many more; can be manipulated to get better results under our dataset.

PARAMETER SELECTION

GridSearchCV of sklearn is used to provide with the best-fit parameters after a full search. It takes an object of model and its parameters as a dictionary, then selects the best from the given list of parameters.

I used the Random Forest Classifier, so I provided the values of its parameters and set `cv = 10`, which gives the best-suited parameters by `best_params_` considering 10 cross-validation sets formed.

The output is entered into the object call of a model with its parameters specified.

The model is then ready to predict the data of `test.csv` using the `.predict()`

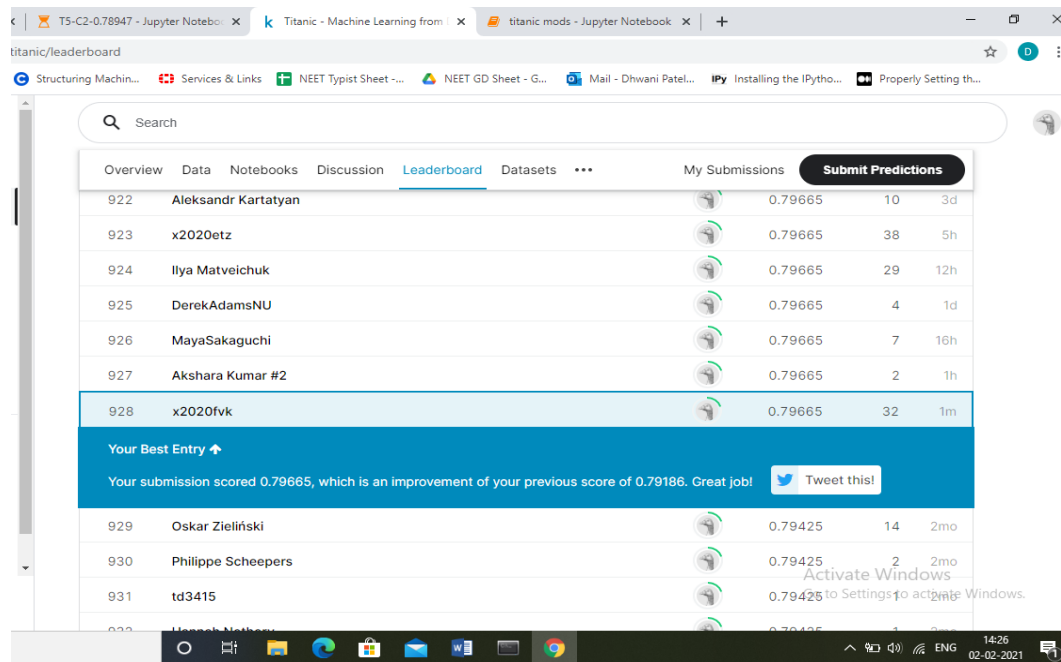
OUTPUT

The trained model predictions for `test.csv` are stored in a file named '`my_submission.csv`' as mentioned on the Kaggle website.

MY ATTEMPTS

I have around 45-50 attempts, below is the list of noticeable changes and the reason behind it.

- 0.76555 – In my first attempt, I used the basics of pre-processing of data and just called the Random Forest Classifier with its default parameters.
- 0.70813 – Then with same pre-processed data, I tried using a simple Decision Tree model with default parameters. But this reduced my score.
- 0.76315 – Used logistic regression model for same pre-processed data.
- 0.77751 – Changed a few default parameters of Random Forest and did scaling of features. Doing this increased my score by a bit.
- 0.78947 – With a few more changes like grouping age for P-class, changing outliers for fare, and some random value of n_estimators for RFC, score increased significantly.
- 0.62918 – I performed binning for both age and fare due to which I got the lowest of all attempts.
- 0.79186 – Removed binning for fare, and used cross_val_score to compare scores and gridsearchcv for best parameter selection.
- 0.79665 – With carefully pre-processing of data like deleting columns, merging columns like SibSp, Parch, etc. and carefully performing gridsearchcv, I achieved my highest score as shown below.



The screenshot shows the Titanic leaderboard on Kaggle. The user x2020fvk is at the top of the list with a score of 0.79665. A blue banner below the table congratulates the user on their best entry.

Rank	Username	Score	Attempts	Time
922	Aleksandr Kartatyan	0.79665	10	3d
923	x2020etz	0.79665	38	5h
924	Ilya Matveichuk	0.79665	29	12h
925	DerekAdamsNU	0.79665	4	1d
926	MayaSakaguchi	0.79665	7	16h
927	Akshara Kumar #2	0.79665	2	1h
928	x2020fvk	0.79665	32	1m
Your Best Entry Your submission scored 0.79665, which is an improvement of your previous score of 0.79186. Great job!				
929	Oskar Zieliński	0.79425	14	2mo
930	Philippe Scheepers	0.79425	2	2mo
931	td3415	0.79425	1	2mo

REFERENCES

- https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- <https://www.kaggle.com/c/titanic/overview>