Capstone Project Report

Sam Stelzner - January 2023

Define **2**
- Overview 2
- Problem Statement 2
- Datasets and inputs 2
- Metrics 3
- Benchmark 3

Analyse **4**
- Data Exploration 4
- Exploratory Visualisation 4

Implement **6**
- Data Preprocessing 6
- Algorithms and Techniques 6
- Refinement 7

Results **8**

Conclusion **9**

Define

Overview

This project is based on the Titanic Kaggle competition. The competition is presented as the best introduction to Kaggle competitions. It was introduced in 2012 by Jessica Li, Will Cukierski. Competition entries from the last two months are presented on a rolling leaderboard. At the time of writing, there were 13,731 teams enrolled. It is a well established problem.

The competition forms part of the field of binary classification. As stated on learndatasci.com, binary classification, in machine learning, is a supervised learning algorithm that categorises new observations into one of two classes. Authors Kumari and Srivastava perform a comprehensive review of work done with binary classification with the aim of detecting sockpuppets. They provide a substantial list of types of classifier algorithms used in this area.

Problem Statement

Disaster struck in 1912 when the RMS Titanic sank after hitting an iceberg. 1502 out of 2224 passengers and crew died. It seems that some groups of people were more likely to survive than others. The problem to be solved here is to predict “what sorts of people were more likely to survive?” based on passenger data. Prediction takes the form of a binary 1 for survived, 0 for deceased.

Datasets and inputs

Train and test datasets have been made available. Train has 891 unique rows, representing passengers. Test has 418. Train includes a target column: Survived. The features of the data sets, with some descriptions and detail of the distribution of the data within these features for the train dataset, taken from Kaggle, are:

- PassengerId - Unique ID
- Pclass - Passenger class (1, 2 or 3)
- Name
- Sex - Male or Female
- Age
- SibSp - Total number of passenger’s siblings and spouse
- Parch - Total number of passenger’s parents and children
- Ticket - Ticket number
- Fare - Ticket price
- Cabin - Cabin number
- Embarked - port of embarkation (Cherbourg, Queenstown or Southampton)
- Survived - 0 for deceased and 1 for survived

Metrics

The evaluation metric is accuracy of correct predictions of whether a passenger survived or not, between 0 and 100%.

Benchmark

This popular model from Kaggle's public notebooks, upvoted 3755 times, can serve as the benchmark. It makes use of the randomForest classification algorithm and has an accuracy of 0.80382.

Analyse

Analysis and implementation are performed using AWS Sagemaker Studio and related AWS tools.

Data Exploration

Train and test datasets are loaded into S3. Sagemaker Studio is used to get a feel for the data and combine into a single dataframe. The summary statistics are shown below. Some insights from this summary include:

- 1309 total rows.
- 891 have a value for ‘Survived’ - these are for all the rows from the original train.csv.
- There are some missing values for ‘Age’ and many missing values for ‘Cabin’. There is one missing value for ‘Fare’ and two missing values for ‘Embarked’.
- The mean age is just under 30. One might think that the Titanic passengers would have been a more wealthy and thus older crowd.

```
[17]: df.describe(include = 'all')
```

[17]:

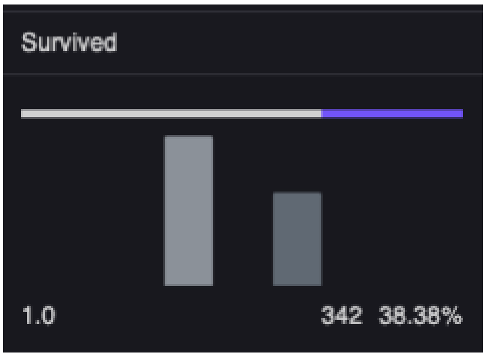
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	1309.000000	891.000000	1309.000000	1309	1309	1046.000000	1309.000000	1309.000000	1309	1308.000000	295	1307
unique	NaN	NaN	NaN	1307	2	NaN	NaN	NaN	929	NaN	186	3
top	NaN	NaN	NaN	Connolly, Miss. Kate	male	NaN	NaN	NaN	CA. 2343	NaN	C23 C25 C27	S
freq	NaN	NaN	NaN	2	843	NaN	NaN	NaN	11	NaN	6	914
mean	655.000000	0.383838	2.294882	NaN	NaN	29.881138	0.498854	0.385027	NaN	33.295479	NaN	NaN
std	378.020061	0.486592	0.837836	NaN	NaN	14.413493	1.041658	0.865560	NaN	51.758668	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.170000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	328.000000	0.000000	2.000000	NaN	NaN	21.000000	0.000000	0.000000	NaN	7.895800	NaN	NaN
50%	655.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN
75%	982.000000	1.000000	3.000000	NaN	NaN	39.000000	1.000000	0.000000	NaN	31.275000	NaN	NaN
max	1309.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	9.000000	NaN	512.329200	NaN	NaN

Dataset summary statistics

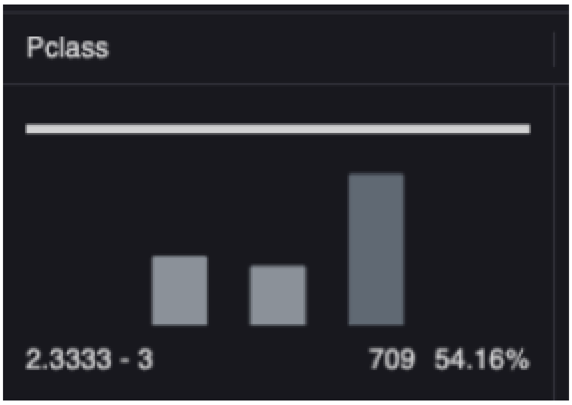
The dataframe is then uploaded back into S3 as dataset.csv.

Exploratory Visualisation

Data Wrangler is used to import dataset.csv. The initial view is helpful for exploratory visualisations. A few are presented below.

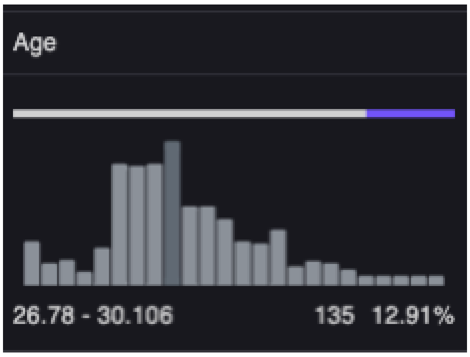


38% Survival Rate



Majority, 54%, with class 3 tickets. This might help explain the younger average age.

Majority male, 64%



Most common age bracket is around 27-30, with a long tail towards the older brackets

Implement

Data Preprocessing

Preprocessing is continued on Data Wrangler. The following transformations are made:

- Missing ‘Age’ and ‘Fare’ values are imputed with the means of those columns
- Missing ‘Embarked’ values are imputed as the most common embarkation value: Southampton (S).
- The index column generated when creating the dataframe and ‘Cabin’, which has many missing values, are dropped

Finally, the dataset is split back into train and test sets and exported to S3 as csv.

Data Wrangler flow

Algorithms and Techniques

AutoGluon is used to discover a good model for this problem. The TabularPredictor method, as used in the code block below, automatically infers a number of useful aspects, such as:

- “AutoGluon infers your prediction problem is: 'binary' (because only two unique label-values observed) - 2 unique label values: [0.0, 1.0]”
- “Note: Converting 1 features to boolean dtype as they only contain 2 unique values.”
- “AutoGluon will gauge predictive performance using evaluation metric: 'accuracy'.”

The fit method in the below block includes a search for the best hyperparameters for each model. All that is needed is to define the time limit in which TabularPredictor will run and that the best quality (the most accurate) model is desired.

```
predictor = TabularPredictor(label="Survived").fit(
    train_data=df_train, time_limit=120, presets="best_quality"
)
```

The best model found by AutoGluon is WeightedEnsemble_L2 with an accuracy of 0.8541.

After some more slight processing, WeightedEnsemble_L2 is then used to make predictions on ‘Survived’ for the test dataset.

```
pred = predictor.predict(df_test)
```

These predictions are extracted from the test dataframe along with passenger IDs and loaded to S3 for submission in the Kaggle competition. The initial results are presented in the following section of this report.

Refinement

To try to improve the predictions accuracy, the TabularPredictor time_limit is tripled to 360 seconds.

With the extra training time, a larger number of models are trialled. However, the ensemble model WeightedEnsemble_L2 is, once again, returned with an accuracy of 0.8541. The predictions made with this new model are identical to the initial predictions.

A further attempt for improvement is to tweak other hyperparameters in aim of improved accuracy, as suggested in the AWS documentation. The following are set:

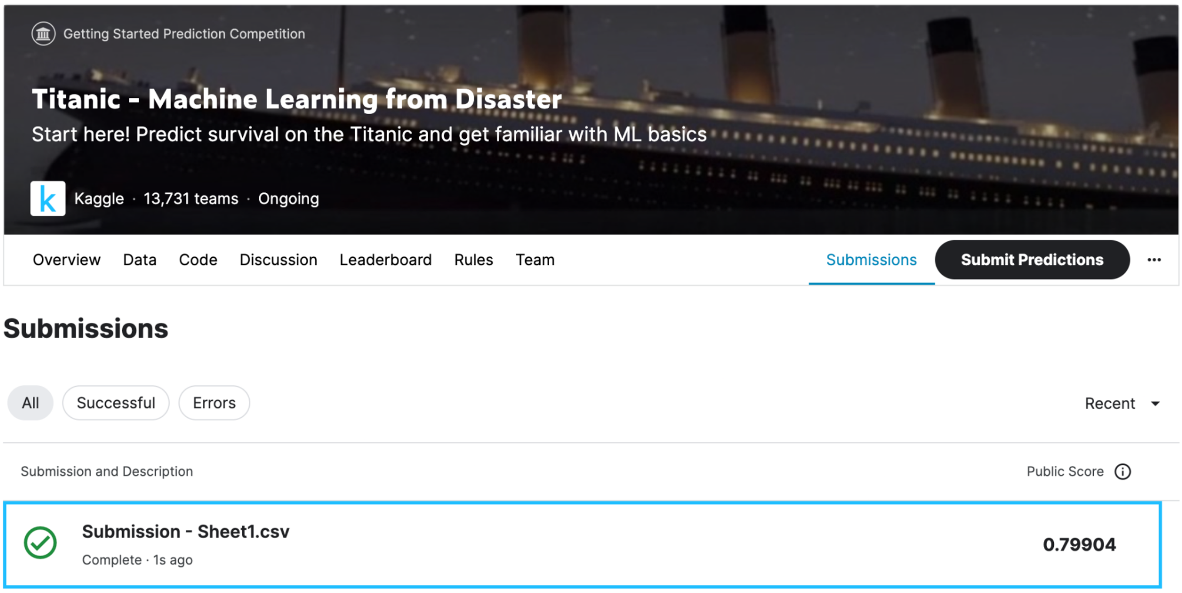
- auto_stack=True
- num_bag_folds = 5
- num_bag_sets = 5

time_limit is also further extended to 500 seconds.

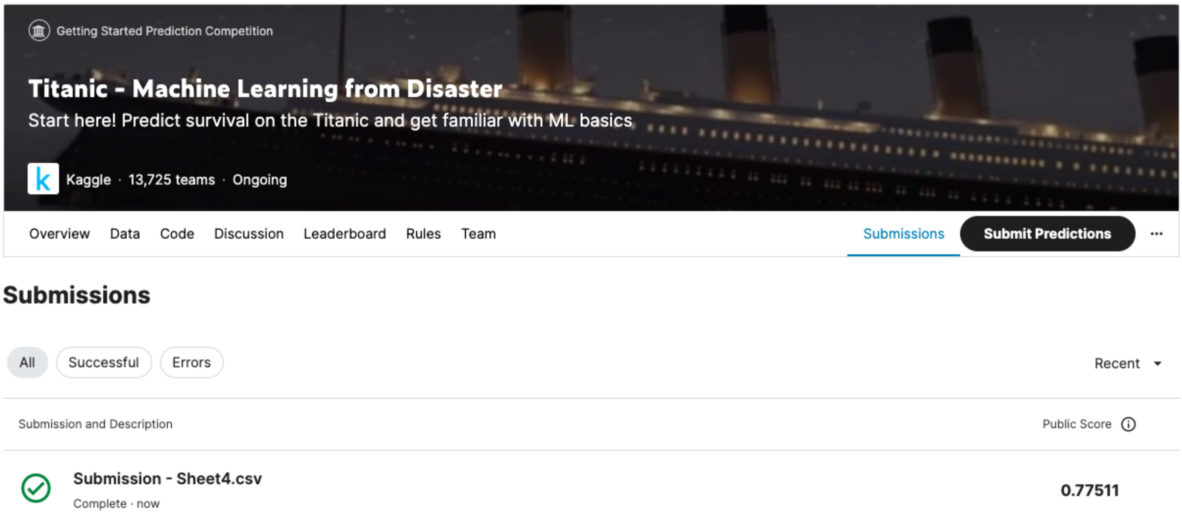
The resulting WeightedEnsemble_L2 model has a slightly improved accuracy of 0.8586. 20 of the 418 predictions made with this new model differ from the previous two. These are submitted to Kaggle.

Results

The figure below shows the initial results when using the WeightedEnsemble_L2 model to make predictions on the test data: an accuracy of 0.79904.



This next figure shows the results of the second submission after tweaking AutoGluon hyperparameters.



Despite the improved accuracy of the trained model, accuracy on the test data is now slightly lower at 0.77511. There is a possibility that the model is overfitting on the trained data.

Conclusion

These results compare admirably to the benchmark model accuracy of 0.80382. Evidently, AutoGluon provides a powerful solution for predictions on this type of problem.

For future projects, opportunities for improvement include further data processing steps such as experimenting with new features built out of the current features and trying individual algorithms with bespoke hyperparameter optimisation, rather than using the built in methods of AutoGluon.