Titanic: Machine Learning from Disaster

hw3

Submission by:

Lorans Ashkar - 315987040

Samar Kardosh - 209319037

Step 1: preparing and organizing datasets and features

1. Reading the dataset from both the train file and the test file.
2. Analyzing the features' types, i.e., numeric, serial and categorical.
3. Changing the "Name" column into a "Title" column according to the person's Title for two main reasons:
    1. We can use them to fill in the missing Age values.
    2. Having more features that can be used in our models might help us build better patterns thus, achieving better result.

Step 11 : preparing and organizing datasets and features (cont.)

4. Checked for missing values and dealt with them accordingly:
   - We filled "Age" column's missing values according to the average of the person's Title (Mr., Mrs., Miss, Sir …)
   - The "Cabin" column had far too many missing values (more than 50%) so we dropped it.
   - The "Embarked" column in the train dataset had only 2 missing values, so we dropped them instead of trying to guess them.
   - The "Fare" column in the test dataset had only on missing value, so we filled it with the column average, using the mean() method.
5. Dropped unnecessary columns ("Ticket" and "PassangerId") because they have unique values and won't contribute to the ML models
6. Made sure that there are no more Nan values.

Step 2: Optimizing the datasets and the models

- Checked the features types. We changed them accordingly to fit all ML models that we used.
- Dropped missing values or filled them – depends on the feature itself – as mentioned in Step 1 point 4.
- We used GridSearchCV() on all models. As well as best parameters and best estimators.

- For Random Forest, we used a method to calculate feature importance and try tried dropping the least important one and checked the affect on the accuracy.
- We used unsupervised clustering on all models to try and improve the results as well.

Algorithms Results

Algorithm (model)	Train and Val. (without clustering)	Train and Val. with Clustering	Test Results (without clustering)	Test - Results (with clustering)
Decision Tree	0.85	0.85	0.794	0.765
Random Forest	0.868	0.868	0.746	0.746
AdaBoost	0.838	0.838	0.717	0.717
SVM	0.82	0.82	0.765	0.765
KNN	0.812	0.823	0.662	0.642
Neural Nets (MLPClassifier)	0.816	0.794	0.755	0.712
PCA	0.835	0.853	0.755	0.765
XGBoost	0.865	0.85	0.746	0.746

Summary

- We assume that the difference between train and test results are due to the fact that the test set is different from the train set , and at some point, our models were over fitted to the train set because we are achieving a Hight accuracy predictions on the train .
- The average of the results of the based trees models was 2% bigger than the average of the other models.
- In most models clustering was not efficient, it gave us the same results .
- In cases where clustering helped , it increased the accuracy approximately 1% .
- High cv values helped us with increasing the accuracy and to avoid overfitting in some cases .
- Even in the models where we included the cluster as a feature , it was not an important feature , and some models advised to drop them to increase accuracy (see optimize_by_droping_features method and Random forest).
- In average we achieved 4.43% accuracy increase after tuning the models (see base model and model after tuning).
- Higher accuracy on train set do not necessarily mean a higher accuracy on test (for example comparing decision tree to rando forest) .
- In models that have a feature importance Parch feature was almost always irrelevant.