RUT-SOM-DATA-PT-06-2020-U-C Douglas High

Machine-Learning-Challenge November 7, 2020

**Summary:** NASA Keppler telescope searches for possible exoplanets and records the raw data. A csv extract of this is found in the data\_in folder. These Keppler Objects of Interest (KOI) are initially given a disposition of ‘candidate’ and then later changed to either ‘confirmed’ (as an exoplanet) or ‘false positive’. Various machine learning models are tested against the known data and then a final model is used to predict against the unknown (‘candidate’) data.

**Programs**

**model0\_compare.ipynb** – tests nine models, each with no scaling, minmax, and standard scaling. Creates a dataframe (and csv) of how each performed with training and testing data. Next, the top three models were chosen and the importance of their features evaluated and then removal of their least relevant features were tested until no negative effects were noticed. Next, GridSearch was performed for select parameters of each top model and the most accurate variations were found.

**model\_top3.ipynb** – the top three model variations were retested and then run against ‘candidate’ records. A csv file is written for each model’s predicted results.

**model\_final.ipynb** – the input\_data is trimmed and tuned to a RandomForest model with no scaling. Model is run against unknown ‘candidate’ records and saved as douglas\_high.sav.

**model1.ipynb** (also **model2**, **3**, **4**, and **5**)- initial programs evaluating training and testing data for various models. Continuation of these programs was forsaken for mode0\_compare.ipynb so I could consolidate their accuracies in a dataframe for evaluation. The only point of interest in these would be model4 which has a list and graph of the nearest neighbors parameter for the KNN model.

**Files**

**data\_in/** **exoplanet\_data.csv** – provided csv file. Extract from NASA file on Keppler Objects of Interest (KOI). koi\_disposition contains status of object with the following three values possible.

1. Confirmed- object has been confirmed as an exoplanet.
2. False positive- object has been confirmed not to be an exoplanet.
3. Candidate- object has not been conclusively identified as an exoplanet or not.

The remaining columns contain data on how the object was measured, its galactic location, temperature, orbital data, etc.

**data\_out/** **methods\_compare.csv** - output from model0\_compare.ipynb. contains a record for each model variation that was tested. This is before any features were removed or model parameters were adjusted. It was used to evaluate the general accuracy of each model variation.

**data\_out/dt.csv** – output from model\_top3.ipynb . contains ‘candidate’ records from data\_in file with features for the DecisionTree model and koi\_predicted, containing ‘confirmed’ or ‘false positive’ depending on the model’s predictions.

**data\_out/rf.csv** – output from model\_top3.ipynb . contains ‘candidate’ records from data\_in file with features for the RandomForest model and koi\_predicted, containing ‘confirmed’ or ‘false positive’ depending on the model’s predictions.

**data\_out/svc.csv** – output from model\_top3.ipynb . contains ‘candidate’ records from data\_in file with features for the Support Vector Machine SVC model and koi\_predicted, containing ‘confirmed’ or ‘false positive’ depending on the model’s predictions.

**data\_out/final.csv** – output from model\_final.ipynb . contains ‘candidate’ records from data\_in file run against the final model.

**douglas\_high.sav**- output from model\_final.ipynb. RandomForest, tuned model.

**Comparison of Models.1**

Initial comparison of each model variation was made against all features with no tuning of model parameters. Processing and evaluation took place within model0\_compare.

1. The **linear regression, ridge, lasso, and elastic net** models performed the worst, with average scores ranging from 70% to 76% accuracy (linear regression with standard scaling averaged -699%?).
2. **KNN** and **logistical** **regression,** both with no scaling, scored in the low 80’s.
3. All other models’ variations (**decision tree, random forest, SVM/SVC, and kNN and logistic regression with scaling**)scored exceptionally well, in the upper 90’s. The only exception being the SVC model with no scaling, which still scored 95% but took about 15 minutes to run (compared to a few seconds for each of the others).
4. The top three results all came from the three decision tree variations, followed by two random forest variations and finally an svm/svc variation. I chose to take the top decision tree (with standard scaling), random forest (no scaling), and svc(standard scaling) models and focus on tuning them.

**Comparison of Models.2**

Importance of features of my top three models were evaluated and the lower end features were dropped. This did not always produce beneficial results. Once satisfied with the features I would use for each model, I ran a gridsearch on selected parameters for each and then did a comparison on how that improved accuracy. Evaluation of accuracy was made against the average of training and testing data and this is the score listed below. The f1 scores for all three after tuning was 99% for ‘confirmed’ objects and 98% for ‘false positive’. Processing and evaluation took place within model0\_compare.

1. **DecisionTree (standard scaling)** – features with a score of 0 were dropped. Elimination of features caused varying results when executed, some with slightly lower scores and sometimes slightly better. Gridsearch produced slight increase in accuracy. Final score was 0.9919.
2. **RandomForest (no scaling)**- features with a score below 1% were dropped. Same results as with Decision Tree features and Gridsearch. Final score was 0.9917.
3. **SVM/SVC** **(standard scaling)**- features with a negative score were dropped. Initially this lowered the accuracy of the model. One by one features were removed from being dropped until all features whose removal lowered the score when dropped were kept and those whose removal had no effect were removed. I also removed features with a score less than 3% which did not lower score, still no increase over all features. In the end the accuracy of the model remained unchanged from when it was run with all features. Final score was 0.9922. Being the highest scoring model, this was initially chosen for my final model.

**Comparison of Models.3**

The SVM/SVC model was chosen and run against the 1687 unknown (‘candidate’) records. The result was a prediction of 3 ‘confirmed’ and 1684 ‘false positive”. This did not seem right to me so I created another program, model\_top3.ipynb, where I ran the top 3 model variations again against the known data and then used them to predict against the unknown. The results were written out as separate csv files.

1. **DecisionTree (standard scaling)** – this model predicted 1665 ‘confirmed’ records and 22 ‘false positive’.
2. **RandomForest (no scaling)**- this model predicted 1515 ‘confirmed’ records and 172 ‘false positive’.
3. **SVM/SVC** **(standard scaling)**- as stated, this model predicted 3 ‘confirmed’ records and 1684 ‘false positive’.

The RandomForest model was then chosen as my final model because it was the least biased. model\_final.ipynb was written using RandomForest. It contains all steps involved; initial training/testing, feature selection (re-training and testing and comparison against none), GridSearch tuning (and comparisons), predictions, and saving of the model.

**Conclusion**

There are obviously many intricacies to machine learning in general as well as for each model type. I’m not sure how I feel about my top three models all doing so well during training and testing (same f1 scores) and yet producing significantly varying results when used to predict unknown data.

In consideration on things I could do to further improve my results…

1. I could go through the list of features and spend more time running tests on how each one affects the model’s accuracy. A simple start would be to one by one drop them and retest the model. Beyond that there is the possibility that the removal or inclusion of groups of features could have and affect. This is something that could be done programmatically, my assumption being this would take quite a bit of CPU time.
2. I could run a more in depth GridSearch, including more parameters as well as more values for those parameters that accepted numeric values. This I know would take up resources and require hours of CPU time.
3. I could have focused on the three variations of DecisionTree since they were the top three performers. I chose to take the top performing variation of the top three models to run more in-depth analysis on, so as to have a more varied test.

In the end I was surprised at how well most models scored in general and that the top performers seemed to produce near perfect results before any tuning. I had to question if I was doing something wrong, which led me to spend time evaluating my code in detail. Ultimately, I am happy with the results and what I have learned but find myself with a slew of questions on machine learning.

One more point of observation. When I was creating/running model\_final I noticed that each time I evaluated the importance of features, I got different results. My choice of dropping lowest scores was based on less than 1% relevance and I got a slightly different list each time I reran it, as seen by the commented ‘cols’ variable in section 60-Feature-Selection.