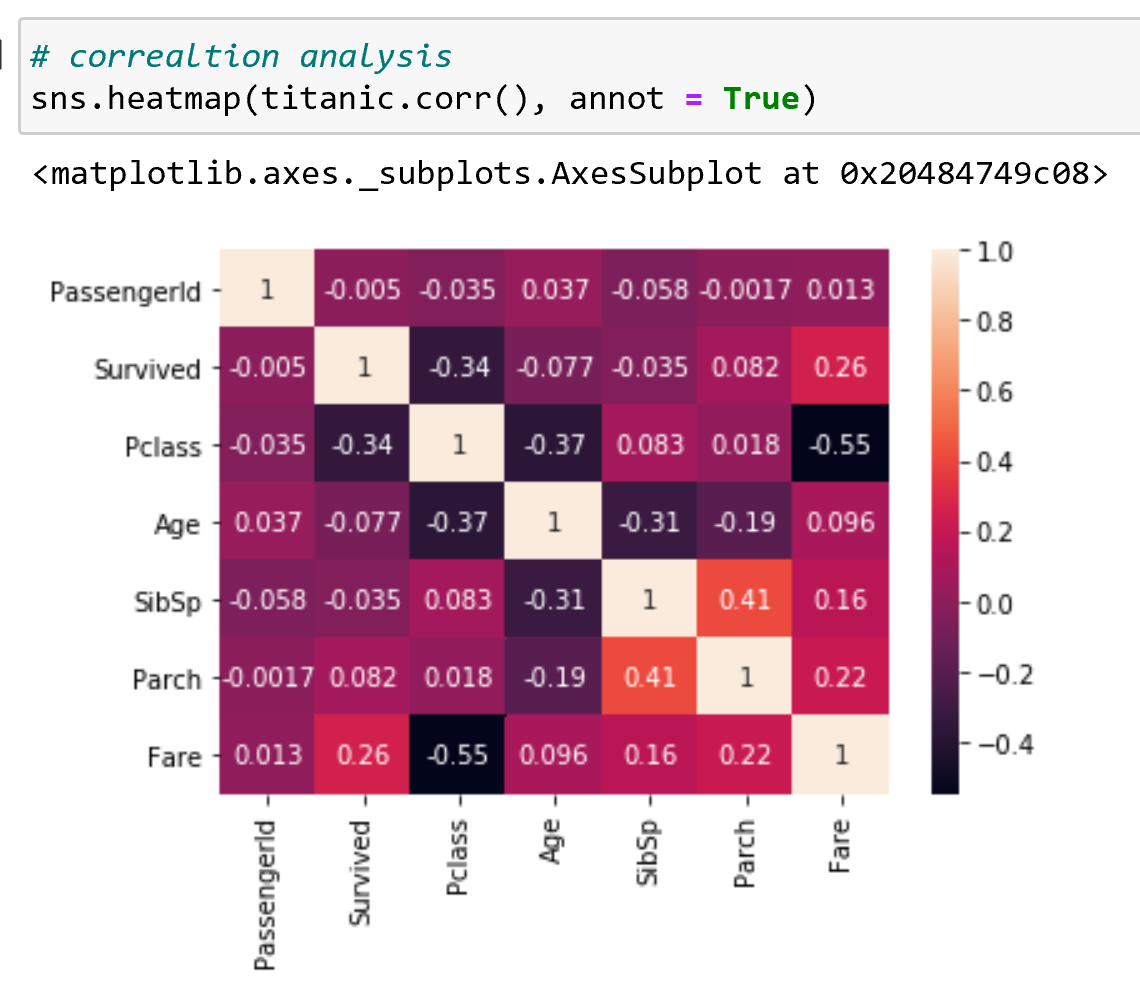
**MEMO**

**Introduction**:

Our task is to create a classification algorithm ensemble that will best predict the outcome of passenger survival on the Titanic. We have confirmed a stacked method with Random Forest, Decision Trees, and Gradient Boosting is the preferred method. We have employed many different classification models and compared their accuracy outputs, continually adjusting parameters in search of increasing model accuracy. You may reference the notebook for EDA and additional explanation. Other than for the sake of interest, this is a meaningful problem as the methods used here to predict survival can be easily translated into the medical field and insurance industries for practical use.

**Data**:

The data used in this project consists of 891 entries (people) with 12 features, all of which were passengers on the Titanic. We transformed the data to consist of only integers, objects, and floating numbers. We dropped unnecessary features that did not add to the predicting power such as “Name”, “Ticket” and “Passenger ID”. The data was sprinkled with “NAN”s in different features. “Cabin” feature had over half of its entries missing, so we decided to not use that in our models, and removed it from the data. There were some minor challenges with our data. We had to transform “Embarked” and “Sex” into objects so they could be properly ran using classification techniques. “Age” had a significant amount of missing values. To combat this, we ran a correlation heat map on all the features, looking for another feature with the strongest correlation to age. “Passenger Class” and “Age” had the highest correlation, and there was a notable difference in the median age for each of the three classes. With this, we then figured the average age in each class. We then filled all the missing values for age as the median age for each individuals’ respective class. Our final data set consisted of 891 data points with 8 features: Survived, Passenger Class, Sex, Age, Number of Siblings and Spouse (combined), Number of Parents and Children (combined), Ticket Fare, and City Embarked. Below, you can see the correlation between all variables.



**Methods**:

When classifying, our output consists of “0”s and “1”s. “0”s representing “Not survived” and “1”s representing “Survived”. Our chosen model consists of an ensemble stack of Random Forest, Decision Tree, and Gradient Boosting classification models. Our preferred Random Forest model was built using the Gridsearch technique. It suggested a max depth of 110, a maximum features of 4, and number of estimators of 50. Our preferred Decision Tree model was built using Gridsearch technique. It suggested a max features of 5, 10 leaf nodes, and minimum sample split of 10. Our preferred Gradient Boosting Classifier was a default, with a cross validation score = 10. On their own, they were each the preferred parameters as they were the highest accuracy when compared to the same models with parameters. Combined, the model returned an out-of-sample accuracy of 83% with a .04% variance at a cross validation score of 10. I arrived at my decisions by first manually adjusting hyperparameters in search of increasing accuracy. I was able to optimize some individual models by employing Gridsearch, and searching for the best parameter tunings. I based all decisions around the accuracy outputs, combined with the precision, recall, and f-1 scores. My motive when tuning hyperparameters was to optimize the accuracy of the model by trying different combinations of hyperparameters.

**Results**:

For each model, I ran the default classifier and used that as a benchmark to guide my hyperparameter tuning.

|  |  |  |
| --- | --- | --- |
| MODEL | Chosen Metric 1 | Accuracy (CV mean) |
| KNN- 1 | KNN = 3 | 68.34% |
| KNN- 2 | KNN = 5 | 71.14% |
| KNN - 3 | KNN = 7 | 69.46% |
| BEST MODEL |  |  |
| KNN -1 | KNN = 5 | 71.14% |

1. I ran three different models with n-neighbor classifiers of 3,5 and 7. The Knn-5 is the default value for KNN models; however it performed the best. We do not have to worry about over fitting with KNN models. I saw a slight increase in accuracy for KNN-5 vs. KNN-3 and KNN-7. There is also an increase in precision and recall statistics for KNN-5 vs. KNN-3 and KNN-7. You can see the output below.
2. **The next models I ran were Decision Trees. I started with the default as a benchmark, then adjusted my parameters from there. To most successfully tune our hyperparameters, we utilized Gridsearch. Gridsearch suggested that we use 5 features, 10 leaf nodes, and a min sample split of 10. When compared to the Decision Tree 2, the CV mean score, and the accuracy score were practically the same. However, the weighted average accuracy increased when utilizing Gridsearch. Our best Decision Tree model is model number 4. After using Gridsearch to find the best parameters, I decided to use those parameters in combination with the AdaBoost Classifier. This increased the strength of the model across the board. Adaboosted Decision Tree with respective parameters is our preferred Decision Tree model. You can see the output below.**

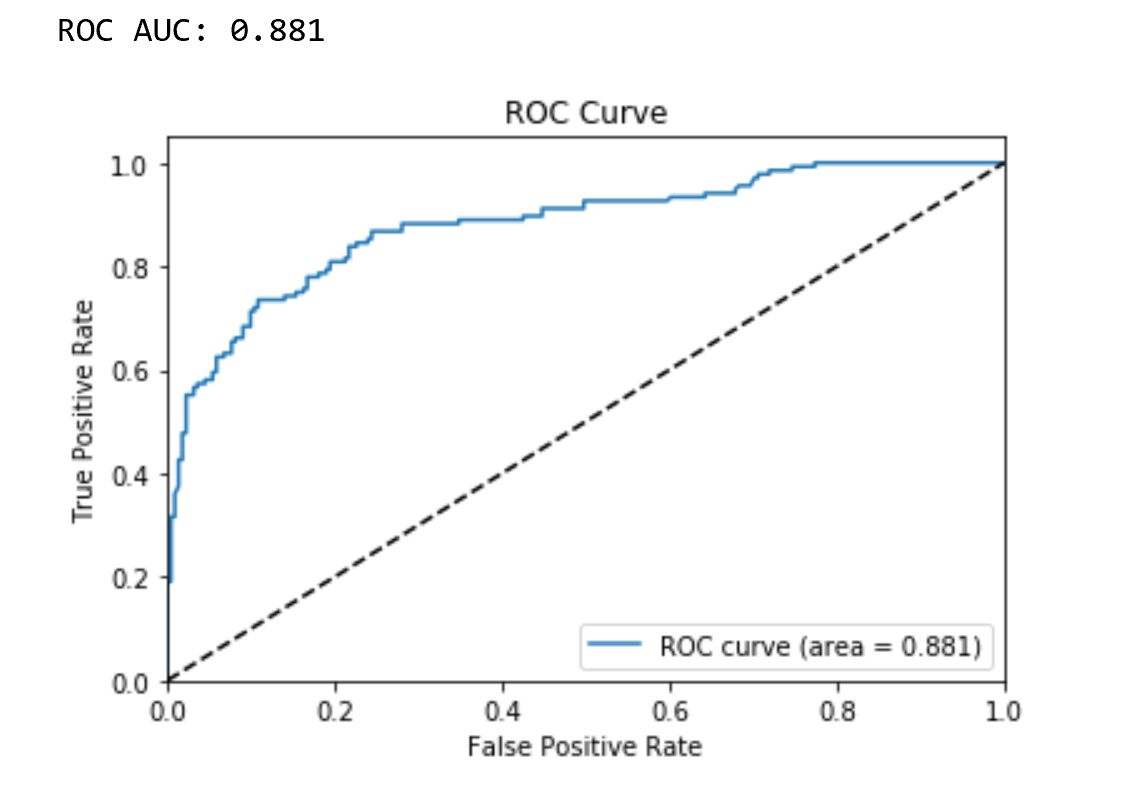
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | Chosen Metric 1 | Chosen Metric 2 | Chosen Metric 3 | Accuracy |
| Decision Tree -1 | default |  | CV = 10 | 73.66% |
| Decision Tree -2 | Leaf nodes = 10 | Weight = balanced | CV = 10 | 80.11% |
| Decision Tree -3 | Leaf nodes = 100 | Weight = balanced | CV = 10 | 78.71% |
| Decision Tree -4 | Grid Search | Features = 5 | Leaf nodes = 10 | 80.67% |
| Decision Tree -5 | Adaboost | Max depth = 3 | Features = 5 | 82.35% |
| BEST MODEL |  |  |  |  |
| Decision Tree -5 | Adaboost | Max depth = 3 | Features = 5 | 82.35% |

1. **We ran 6 Random Forest models. After creating a Random Forest default model, the accuracy is ~ 80%, with precision, recall, and f-1 scores at 80%, 79%, and 80% respectively. We continued to adjust hyperparameters in search of increasing these metrics. After increasing the number of bootstrapped trees to 500, jobs =-1, and oob score = true, the model returned a very similar output, however slightly decreasing all accuracy statistics. After changing the number of bootstrapped trees to 100, jobs =-1, and oob score = true, the model returned a very similar output to the default, however slightly decreasing accuracy and recall statistics. After using Gridsearch to find the best hyperparameters, we were able to increase our overall accuracy, and accuracy statistics to an accuracy of 82%. We can now inspect what features add the most strength to the strongest model:** 
   1. Features sorted by their score: Higher the more important [(0.4755, 'Sex'), (0.1718, 'Fare'), (0.1584, 'Age'), (0.1051, 'Pclass'), (0.0449, 'SibSp'), (0.0299, 'Embarked'), (0.0145, 'Parch')]

**Here, we can again see that 'sex' is the biggest contributor to whether an individual survived on the Titanic.** **After employing Adaboosting Classifier on the default Random Forest, and the best random forest, the accuracy still does not exceed that of the RF Best model. The RF best model is the preferred Random Forest model number 4. You can see the output below.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | Chosen Metric 1 | Chosen Metric 2 | Chosen Metric 3 | Accuracy |
| Random Forest -1 | default | CV = 10 | 78.14% | 80.67% |
| Random Forest -2 | Estimators = 500 | Jobs = -1 | Oob score = true | 79.72% |
| Random Forest -3 | Estimators = 100 | Jobs = -1 | Oob score = true | 79.83% |
| Random Forest -4 | Gridsearch | Depth = 110 | Estimators = 50 | 82.07% |
| Random Forest -5 | Adaboost | “SAMME” |  | 79.55% |
| Random Forest -6 | Adaboost | “SAMME” | RF4 | 77.59% |
| BEST MODEL |  |  |  |  |
| Random Forest -4 | Gridsearch | Depth = 110 | Estimators = 50 | 82.07% |

**We can visualize the strength of our RF Best model by plotting a ROC curve:**



**This ROC curve represents the lift that is present in the Random Forest model. With an AUC value reaching close to 1.0 at .881, this acts as an excellent classifier.**

1. **We ran a total of 3 SVM models on the data. Here, we can see that the default returns a 75% accuracy. We then adjusted the hyperparameters in search of increasing accuracy statistics. When changing the weight classes to balanced, we were able to increase the overall accuracy to 77%, along with recall and f1-score. The precision decreased from 78% to 77%. We then searched to increase the maximum iterations in search of increasing accuracy statistics. After increasing the maximum iterations up to 2500, from the default 1000, we were not only able to increase the overall accuracy of the model, but we were also able to bring the precision statistic back up, to 79%, and also increase the recall and f-1 score to 78%. This surpasses the previous manual adjustments in the last notebook. SVC linear model 3 is our preferred SVC model.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MODEL | Chosen Metric 1 | Chosen Metric 2 | Chosen Metric 3 | Chosen Metric 4 | Accuracy |
| SVC -1 | default | N/A |  | Type = linear | 75.91% |
| SVC -2 | Penalty = L2 | N/A | Weight = balanced | Type = linear | 77.03% |
| SVC -3 | Penalty = L2 | Max iterations = 2500 | Weight = balanced | Type = linear | 77.87% |
| BEST MODEL |  |  |  |  |  |
| SVC -3 | Penalty = L2 | Max iterations = 2500 | Weight = balanced | Type = linear | 77.87% |

1. **We ran a total of two SVC Radial Basis Function models. The default RBF returns a low accuracy of ~ 68%. We searched to increase accuracy by tuning hyperparameters. We were able to optimize SVC RBF accuracy and recall statistics by employing Gridsearch and finding the best possible tunings for our parameters. After using Gridsearch to find gamma, cost function, and degree, we were able to increase our accuracy significantly to over 77%.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | Chosen Metric 1 | Chosen Metric 2 | Chosen Metric 3 | Accuracy  (Training Data) |
| SVC RBF -1 | default |  |  | 67.59% |
| SVC RBF -2 | Type = Kernel RBF | Cost function = 100 | Gamma = .001 | 77.31% |
| BEST MODEL |  |  |  |  |
| SVC RBF -2 | Type = Kernel RBF | Cost function = 100 | Gamma = .001 | 77.31% |

1. **We ran a total of 5 artificial neural network employing stochastic gradient descent models. With defaults (“Relu”, “Adam”), this ANN model employing a stochastic gradient descent solver (“Adam”) returns an accuracy of 73% when containing 7 neurons (n-1 features) at 3 levels. We continued to adjust the number of neurons and layers to increase accuracy. For the second model, decreasing the number of layers resulted in a decrease of accuracy when compared to default. For the third model, increasing nodes and layers resulted in an increased accuracy. For the fourth model, the continued increasing of node sizes results in increased accuracy. After finding that increasing node sizes and number of layers increased accuracy, we found the model plateaued after 10 nodes of 3 layers, resulting in an accuracy of ~ 80%. Our preferred ANN model consists of 10 nodes at 3 layers.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MODEL | Chosen Metric 1 | Chosen Metric 2 | Chosen Metric 3 | Chosen Metric 4 | Accuracy |
| ANN -1 | Layers = 3 | Nodes = 7 | Activation = Relu | Solver = Adam | 73% |
| ANN -2 | Layers = 2 | Nodes = 7 | Activation = Relu | Solver = Adam | 72% |
| ANN -3 | Layers = 3 | Nodes = 8 | Activation = Relu | Solver = Adam | 80% |
| ANN -4 | Layers = 3 | Nodes = 10 | Activation = Relu | Solver = Adam | 80% |
| ANN -5 | Layers = 3 | Nodes = 20 | Activation = Relu | Solver = Adam | 79% |
| BEST MODEL |  |  |  |  |  |
| ANN -4 | Layers = 3 | Nodes = 10 | Activation = Relu | Solver = Adam | 80% |

**7,8,9. Other Classifiers**

**I ran three Bagging classifiers on the data. The default retuned an accuracy of 78%. After adjusting the number of estimators, oob score = true, n\_jobs =-1, the model was able to increase accuracy and recall statistics. We then increased the number of estimators to 1000. This only increased the run time of the model, and did not result in an increased accuracy. Our preffered bagging model consists of 100 estimators, n\_jobs =-1, oobscore = true. You can see the output below.**

**Our default Gradient Boosting classifier returned an accuracy and all recall statistics at 81%. We then worked to turn hyperparameters in search of increasing accuracy. After adding estimators and learning rate, we were not able to increase our accuracy. I ran a total of 5 Gradient Boosting models on the data. The default returned an accuracy of 81% and all recall statistics of 81% as well. We looked to adjust hyperparameters in search of increasing accuracy. After adjusting the learning rate and number of estimators, we were unable to increase our accuracy. Our preferred Gradient Boosting Classifier was the deafult with max\_features=1.0, max\_samples=1.0, n\_estimators=10, n\_jobs=None, oob\_score=False, random\_state=None, verbose=0 at 80.95% accuracy. You can see the output below.**

**The Extra Trees Classifier default returned an accuracy of 78%, along with recall statistics equaling 78% as well. We then adjusted the hyperparameters in search of increasing accuracy and recall statistics. After the 3rd model, it was apparent that decreasing the depth resulted in a decrease in accuracy. In total, we ran five Extra Trees Classifier models on the data. The default returned a 78% accuracy. We then tuned hyperparameters in search of increasing accuracy. After increasing max depth, and decreasing estimators back to the default 100, we were able to increase our accuracy and recall statistic to 81%. You can see the output below.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MODEL | Chosen Metric 1 | Chosen Metric 2 | Chosen Metric 3 | Chosen Metric 4 | Accuracy |
| Bagging -1 | default | N/A | N/A | N/A | 78.22% |
| Bagging -2 | # estimators = 100 | Oob = true | N\_jobs = -1 | N/A | 80.23% |
| Bagging -3 | # estimators = 1000 | Oob = true | N\_jobs =-1 | N/A | 80.11% |
| Gradient Boosting -1 | default | N/A | N/A | N/A | 81.34% |
| Gradient Boosting -2 | # estimators = 500 | CV = 10 | Random State = 0 | Depth = 3 | 79.01% |
| Gradient Boosting -3 | # estimators = 500 | Learning rate = .5 | Random State = 0 | Depth = 3 | 79.27% |
| Gradient Boosting -4 | # estimators = 500 | Learning rate = .1 | Random State = 0 | Depth = 3 | 79.27% |
| Gradient Boosting -5 | # estimators = 1000 | Learning rate = .1 | Random State = 0 | Depth = 3 | 79.83% |
| Extra Trees -1 | default | N/A | N/A | N/A | 77.87% |
| Extra Trees -2 | N/A | Weight = balanced | Depth = 5 | N/A | 79.27% |
| Extra Trees -3 | N/A | Weight = balanced | Depth = 3 | N/A | 78.71% |
| Extra Trees -4 | Estimators = 1000 | Weight = balanced | Depth = 5 | N/A | 78.99% |
| Extra Trees -5 | Estimators = 500 | Weight = balanced | Depth = 10 | N/A | 80.67% |
| BEST MODEL |  |  |  |  |  |
| Gradient Boosting -1 | default | N/A | N/A | N/A | 81.34% |

**10. Ensemble stacking**

**When stacking our best Random Forest, Adaboosted Decision Tree, and best SVM, model, the stack returns an accuracy of 82% (+/- .02) at a CV = 5. We then worked to combine different models and tune parameters in search of increasing accuracy. When stacking our best Random Forest, Adaboosted Decision Tree, and best SVM, model at a CV =10 the stack returns an accuracy of 82%(+/- .04). The increase in the cross-validation number resulted only in an increase in the variance, which can be expected when adding more validations. For the third model, we decided to do away with the SVM, and add in the best Gradient Boosting Classifier. When stacking our best Random Forest, Adaboosted-Decision Tree, and Gradient Boosting Classifier at CV =5, the stack returns an accuracy of 82%. We then increased the CV equal to 10, and ran the same model in search of increasing accuracy. After increasing the cross-validation score, the stack returned a lesser accuracy and higher variance. I ran a total of 4 models that consisted of a stack of many different previous models. The first model consisted of Random Forest, Decision Tree, and SVM model 3. Individually, these models were the best of their kind. Combined, they returned 82% accuracy. I then increased the cross-validation score of the same ensemble, and the accuracy was stagnant at, 83% however it also increased the variance. For the third model, I combined the best Decision Tree, Random Forest, and Gradient Boosting models. This returned an ensemble accuracy of 82%. I then tried to increase the cross-validation score to CV = 10 in search of increasing accuracy in my 4th model. This resulted in an increase in accuracy, and a slight increase in variance. The stack number 4 is the preferred model of all models created at an accuracy of 83% (+/- .04% variance), composed of random forest, decision tree, and Adaboosted Decision Tree. Ensemble model 4 is all around the best performer, and I recommend that it be used when searching for the proper survival classification of passengers on the Titanic.**