Vincent La

903178639-vla6-hw1

# Descriptive Statistics

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Deceased Patients | Alive Patients | Function to Complete |
| Event Count |  |  | Event\_count\_metrics |
| 1. Average Event Count | 982.014 | 498.118 |
| 1. Max Event Count | 8635 | 12627 |
| 1. Min Event Count | 1 | 1 |
| Encounter Count |  |  | Encounter\_count\_metrics |
| 1. Average Encounter Count | 23.038 | 15.452 |
| 1. Max Encounter Count | 203 | 391 |
| 1. Min Encounter Count | 1 | 1 |
| Record Length |  |  | Record\_length\_metrics |
| 1. Average Record Length | 127.532 | 159.2 |
| 1. Max Record Length | 1972 | 2914 |
| 1. Min Record Length | 0 | 0 |

# Model Performance

**Model Performance on Training Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **AUC** | **Precision** | **Recall** | **FScore** |
| Logistic Regression | 0.9545454545454546 | 0.9454047619047619 | 0.9869281045751634 | 0.8988095238095238 | 0.9408099688473521 |
| SVM | 0.9940191387559809 | 0.9945119047619048 | 0.9882005899705014 | 0.9970238095238095 | 0.9925925925925925 |
| Decision Tree | 0.7763157894736842 | 0.7475952380952382 | 0.792156862745098 | 0.6011904761904762 | 0.6835871404399323 |

**Model Performance on Test Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **AUC** | **Precision** | **Recall** | **FScore** |
| Logistic Regression | 0.7380952380952381 | 0.7375 | 0.6804123711340206 | 0.7333333333333333 | 0.7058823529411765 |
| SVM | 0.7380952380952381 | 0.7388888888888889 | 0.6767676767676768 | 0.7444444444444445 | 0.708994708994709 |
| Decision Tree | 0.6714285714285714 | 0.6569444444444444 | 0.6329113924050633 | 0.5555555555555556 | 0.591715976331361 |

Based on the performance on the training vs test data, we can clearly see that the linear models seem to be overfitting the training set as overall accuracy, AUC, Precision, Recall Fscore for the linear classfiers (Logistic Regression and SVM – remember that we used a linear SVM) are all very high for the training set but for the test data performance is much worse.

However, for the decision tree model, which is non-linear we can see that the training performance is worse, but the test data performance is not as steep of a drop-off in performance. Thus, more training data and better parameter tuning may help.

One thing that may also help is to reduce the dimensionality of this data set. Since the data is very wide, but sparsely filled in, there might be improvements if we used clinically meaningful ontologies to create higher level groupings. For example, similar drugs can be grouped together if they share the same therapeutic class. This will reduce the variance in the model. \

# Model Validation

|  |  |  |
| --- | --- | --- |
| CV Strategy | Accuracy | AUC |
| K-Fold | 0.7213216424294269 | 0.7075773303028468 |
| Randomized | 0.7357142857142858 | 0.7188220160244053 |

# Creating Own Model

For my own model, I decided to use a Random Forest model. The reason why I wanted to use a Random Forest model is that I saw in earlier parts that linear models seemed to overfit the training data. Notice that in previous parts, the Logistic Regression and Linear SVM had very high Training data performance, but performance was severely worse in Test data performance. However, we saw that in the Decision Tree, while overall performance was worst, it did not overfit as much.

Thus, I thought a Random Forest model would do better since as a Tree-based model it could deal well with sparse data, but it would not overfit like a Decision Tree would.

One thing I wanted to do was tune the number of trees in the Random Forest. This can be done by tweaking the “n\_estimators” parameter of RandomForestClassifier in Sci-Kit Learn. What I did was I took the training data set, and then used **K-Fold CV Strategy** and report the Accuracy and AUC below:

|  |  |  |
| --- | --- | --- |
| Number of Trees (n\_estimators) | Accuracy | AUC |
| 10 | 0.6590034217279725 | 0.6471645199125903 |
| 50 | 0.7045124037639008 | 0.7076620404436934 |
| **100** | **0.7176860564585116** | **0.7245148751481655** |
| 200 | 0.7045052751639578 | 0.7103113374829012 |
| 300 | 0.7021314513829483 | 0.7089828350674422 |

As you can see, it looks like based off K-Fold validation, using 100 trees is optimal. Yes, this does slightly better than the Logistic Regression performance reported in the previous section where AUC was ~0.707 compared to 0.725 with the random forest with 100 trees.