Predicting Readmission in Patients with Diagnosed DM

Andrew Burton, RN, BS, BSN

Background

Frequent patient readmissions are taxing on hospitals and indicative of quality of care. With hospitals being chronically short-staffed and lacking sufficient beds, ensuring a discharged patient not be readmitted is imperative in managing limited healthcare resources.

Additionally, there is financial incentive to reduce readmission rates – under the Hospital Readmissions Reduction Program (part of the Affordable Care Act), Medicare reimbursement to hospitals is penalized in cases of readmission within 30 days of discharge.

Because of this, there is strong motivation for hospitals to be able to predict if a patient will be readmitted within 30 days of their discharge. Given the troves of data collected by our healthcare system and the power of machine learning algorithms, it is possible we can predict which patients are likely to be readmitted. With this information, providers can make better-informed decisions on when and where a patient with be discharged.

Research Question

> *Can machine learning algorithms be used to identify patients that are likely to be readmitted within 30 days of hospital discharge?*

Data

To train the model, I used a dataset of hospital admissions assembled by Strack et. al. The data spans from 1999 to 2008 and was collected from 130 hospitals. Altogether, the data has about 101,000 records of hospital admissions. The following inclusion criteria were required: diagnosis of DM in chart, at least one lab obtained during visit, at least one procedure performed per visit, and a length of stay between 1 and 14 days.

The target variable used was patients' outcomes in terms of readmission, with outcome being a three-level nominal variable. Specifically, each entry was labeled by one of; readmitted in less than 30 days, readmitted after 30 days, or never readmitted. For the sake of classification, the target variable was mapped to 'readmitted within 30 days of discharge' or 'not readmitted within 30 days of discharge.'

Records in the dataset had 54 other attributes that were used features in model training.

Identifiers

Patient ID, Encounter ID (neither used as features)

Numeric

Total number of days admitted
Number of ER visits, inpatient visits, outpatient visits within one year prior to admission
Number of procedures, lab procedures performed during visit
Number of distinct medications administered during visit
Number of diagnoses recorded in chart

Categorical

Race (5 levels), Gender (2 levels), Age (Binned, 10 levels)
Weight (Binned, 9 levels)
Admission Type/Source (9 and 21 levels, respectively)
Discharge Disposition (29 levels)
Medical Specialty (84 levels)
Payer Code (23 levels)
Primary/Secondary/Tertiary Diagnoses (ICD10 codes. >900 levels)
Highest serum glucose level (binned, 4 levels – not tested, normal, >200, >300)
HbA1c result (binned, 4 levels – not measured, not tested, <7%, 7-8%, >8%)
Change in dose of 24 different DM meds (4 levels – none, steady, increased, decreased)
Change in dosing of any glycemic-control med (2 levels)
Visit payer source (23 levels)
Medical specialty of admitting provider (84 levels)

Bias

Healthcare has a noted history of focusing research on white males and exercising discrimination towards minorities, women, and LGBTQ persons (to put it lightly). This dataset does evidence that – people of color are underrepresented in the dataset and gender labels are limited to binary male/female (fig. 1). It is important to be mindful of this when considering the results of modeling as performance in predicting the outcomes of these underrepresented populations may be impaired or skewed. Females were slightly more prevalent than males in the data (fig. 2).

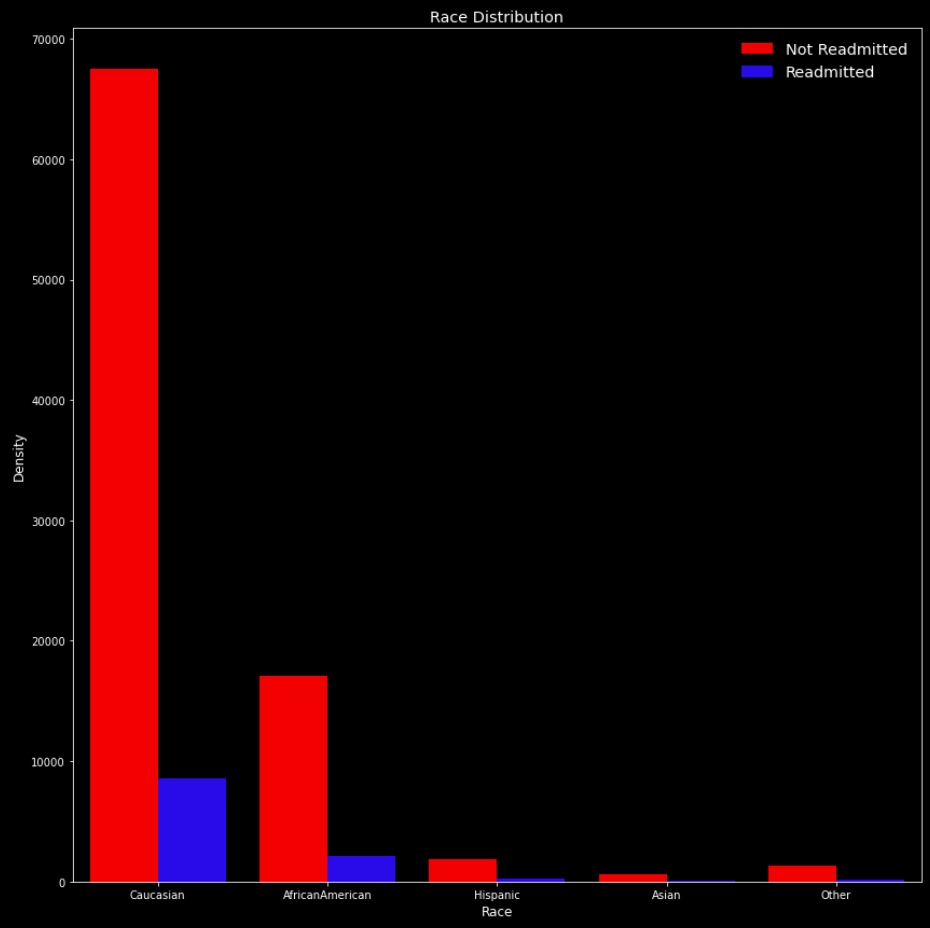

Figure 1: Race Representation in Dataset

Data Preparation

Cleaning

There were about 3000 records that did not have a recorded race, seemingly completely at random. Imputing these values would be problematic, so all records missing the patient's race were dropped. Additionally, there were a few records missing a primary diagnosis that were dropped as well, largely because I was going to use this column in the imputation of other missing values. Finally, one row that had gender recorded as 'unknown/invalid' was dropped. After removing the above, the dataset had 99,473 rows remaining.

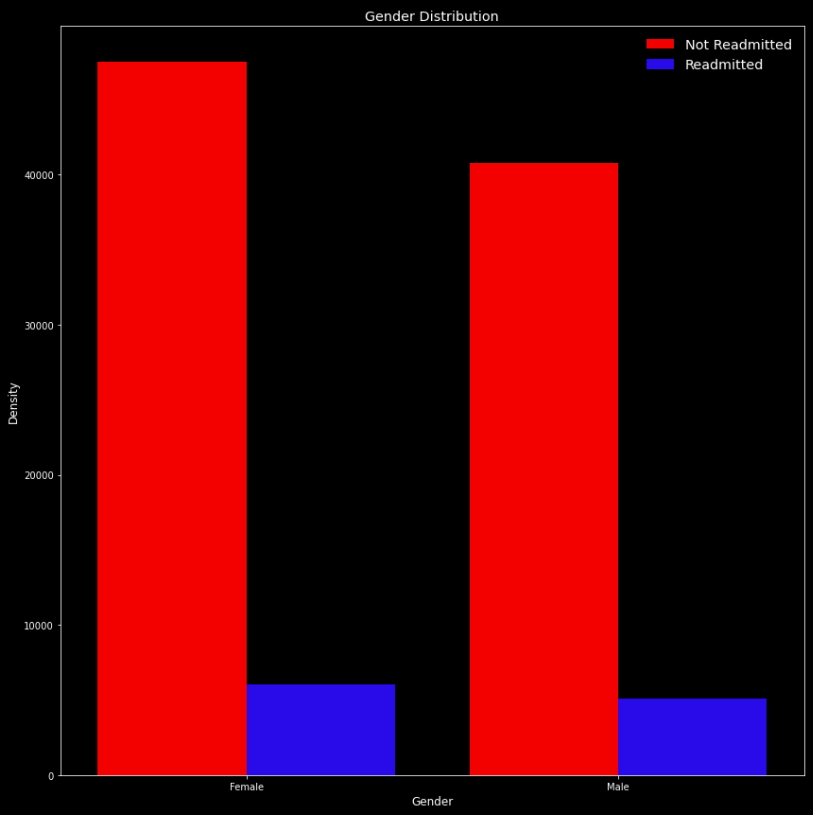

Figure 2: Gender Representation in Dataset

Values were missing from the secondary and tertiary diagnoses. It is not unusual to have only one diagnosis during a hospital visit, so these missing values were filled with 'none,' treated as an additional categorical level. Because I did not think imputing what entity paid for each visit would prove to be useful, the same was done for payer code attribute.

Two additional columns had a large number of missing entries, the first of which was the medical specialty that managed the patient's care during the visit. For this attribute, about 50% of values were missing. I thought it reasonable to infer this value based off of the patient's primary diagnosis.

Before I could do this, I needed to address the fact that there were over 60 levels for medical specialty and over 900 unique ICD10 codes recorded as diagnoses. For both these attributes, I amalgamated into more general classes. For example, all possible gi neoplasms diagnoses were grouped together, and all surgical specialties were treated as the same class. This reduced the number of levels for diagnoses (primary, secondary, and tertiary) to about 50 and the number of levels for medical specialty to 16.

After this, I mapped the missing medical specialty values to the specialty that would likely be managing their care based off their primary diagnosis.

The final column missing values was weight, which was not recorded in about 90% of instances. Because I thought weight would have a good degree of explanatory power, I thought it important to impute these missing values, rather than drop the column. To do so, I used multiple imputation (specifically MICE) to replace missing weights based off the patient's age, gender, and primary diagnosis. Better results might be possible using additional features, but performance was an issue, especially given the number of missing values.

Feature Engineering

After addressing missing values, I converted the two ordered categorical variables (weight and age) into numeric, assigning the upper end of each bin as an integer to its respective instance.

Finally, several additional features were added to the dataset. The average number of procedures, lab procedures, and medication per day was calculated for each instance. Using the primary diagnosis, I created a column with any additional complications upon admission, either: diabetic ketoacidosis, hyperosmolality, renal, neurological, vascular, or none. Finally, using all the various medication dose changes columns, I added a two-level attribute reflecting if there DM was treated with medication prior to admission (patients with 'none' in all medication columns).

To prepare the data for modeling, all categorical features were converted into dummies with the most common level removed.

An 80%, 10%, 10% split was used for the training, validation, and test sets, respectively. All numeric features were scaled using a standard scaler fit to the training set. Because the target outcome 'admitted before 30 days' was underrepresented in the data, I used SMOTE to oversample the data, creating more instances of both 'admitted before 30 days' and 'admitted after 30 days', resulting in a training set of about 140,000 instances (fig. 3).

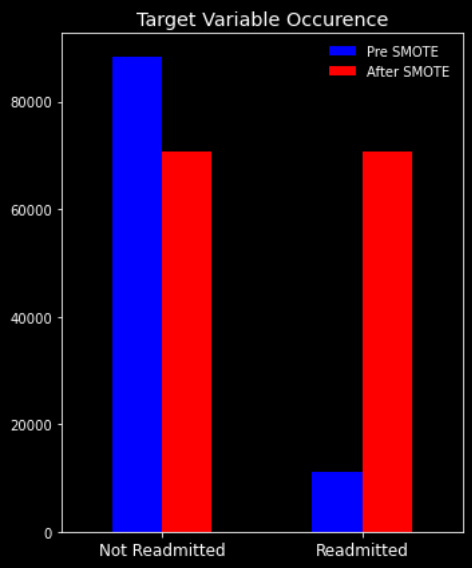

Figure 3: Target Variable Occurrence Before and After SMOTE

Modeling

Model Selection

Five different classification models were trained and evaluated based off of their performance on the validation set: KNN, Logistic Regression, Naïve Bayes, Neural Network, and Random Forest. Overfitting was an issue for all models except the logistic regression, having a drop in performance on the validation set compared to the train set (table 1, fig 3, Appendix A).

The results were impacted by the class imbalance – many achieved a high accuracy by over-predicting 'no readmission.' A model that failed to predict any readmissions would have an 89% accuracy. Because of this, I wanted to use another metric for evaluating performance. For this reason, I favored a lower false negative rate and AUC over accuracy.

The KNN and naïve Bayes models had the lowest false negatives rates but at the expense of high false positives. Ultimately, I chose to proceed using a logistic regression. While this model's accuracy wasn't as good as that of the neural network and random forest, it had the best balance of false negatives and false positives. Additionally, The AUC score of the logistic regression was the best of the 5 models.

	Accuracy		**Precision**		**Recall**		**False Pos**		**False Neg**		**AUC**
	Train	Validate	Train	Validate	Train	Validate	Train	Validate	Train	Validate	
KNN	0.78	0.46	0.70	0.13	0.99	0.68	30,565	5,033	10	365	0.57
Logistic Regression	0.65	0.63	0.65	0.16	0.65	0.53	24,313	3,102	25,367	523	0.64
Neural Network	0.99	0.83	0.99	0.18	0.98	0.14	64	746	1,697	972	0.58
Naïve Bayes	0.55	0.21	0.53	0.12	0.97	0.91	61,705	7,736	2,423	99	0.52
Random Forest	1.0	0.82	1.0	0.18	1.0	0.16	0	823	0	947	0.59

Table 1: Model Selection Results

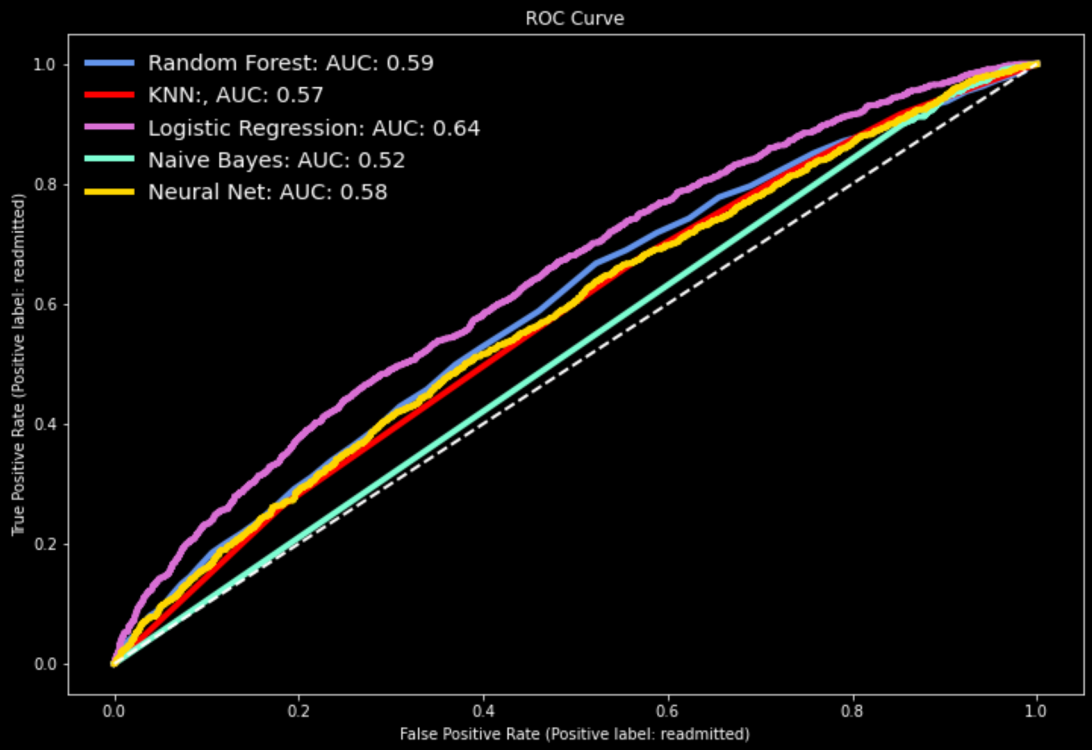

Figure 4: Model Selection ROC Curve

Feature Selection

Proceeding with the logistic regression, I then attempted to optimize the input features, with and without PCA. Based off the coefficients' magnitudes from the base model, I used a forward selection approach to select features (fig. 5, table 2, Appendix B). The feature sets that yielded the best accuracy, precision, and recall chosen as candidates. The model with the best accuracy utilized the 8 features with the largest coefficient magnitude, the top 100 features yielded the best precision, and the best recall was achieved with 1 feature.

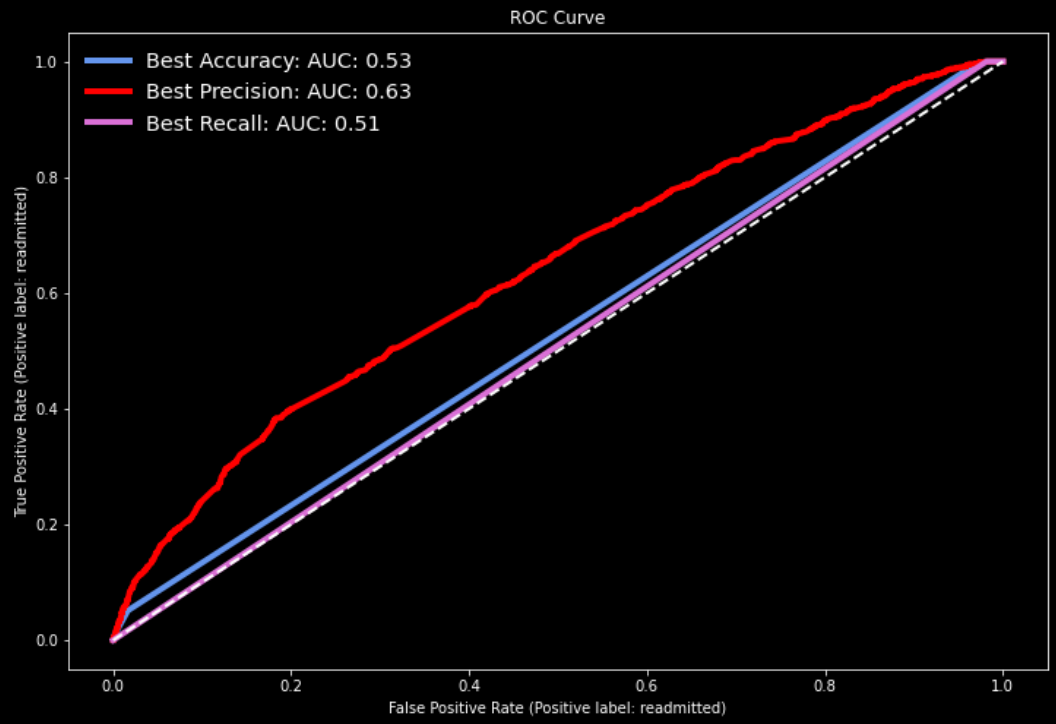

Figure 5: Forward Feature Selection ROC Curve

This process was repeated using PCA – I iterated through number of principle components pulled from the dataset. I then compared the models that yielded the best accuracy, precision, and recall, which used 10, 290, and 170 principle components, respectively (fig 6, table 2, Appendix C). Looking at the cumulative variance curve, 10 components account for about 10% of the variance in the data and both 170 and 290 components account for over 99% (fig 7).

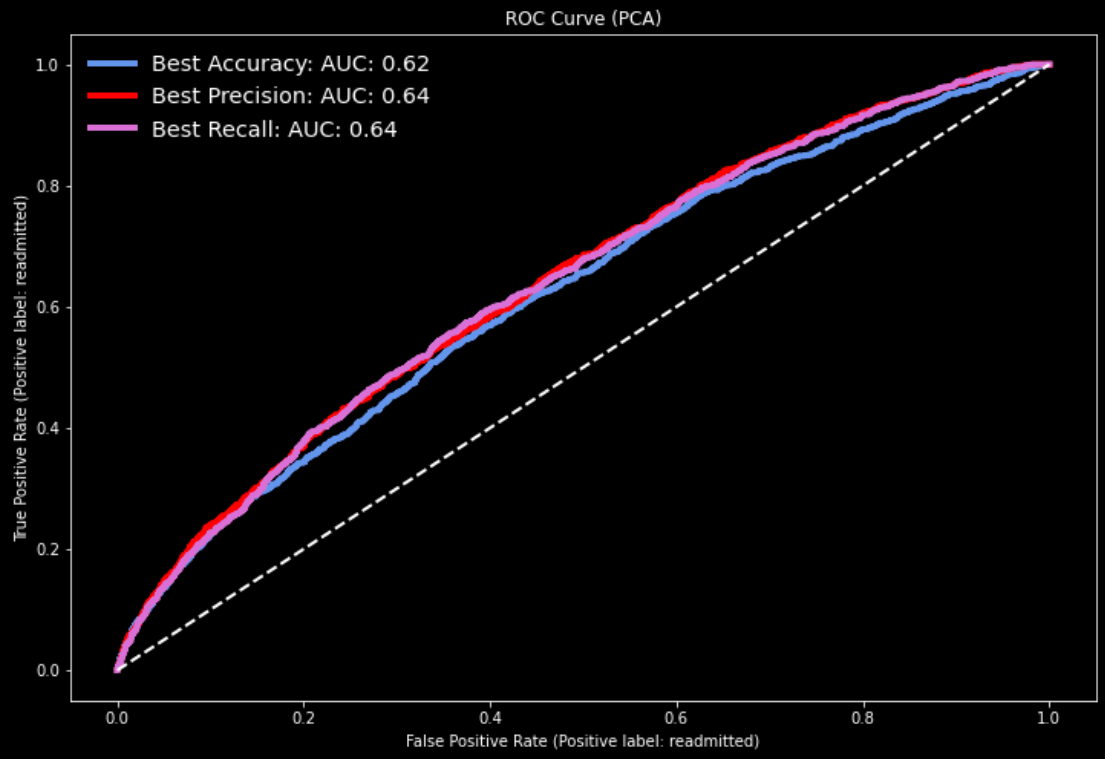

Figure 6: PCA Results

The comparative performance of the six candidate models was then evaluated against each other. The models that used 8 and 1 features performed the worst, with an AUC of around 0.5. The remaining four models had similar performances in terms of their AUC scores, being around 0.63. The number of false negatives for these models was similar as well. Because it had the lowest false positives, I chose to proceed with the model that used 100 features without PCA.

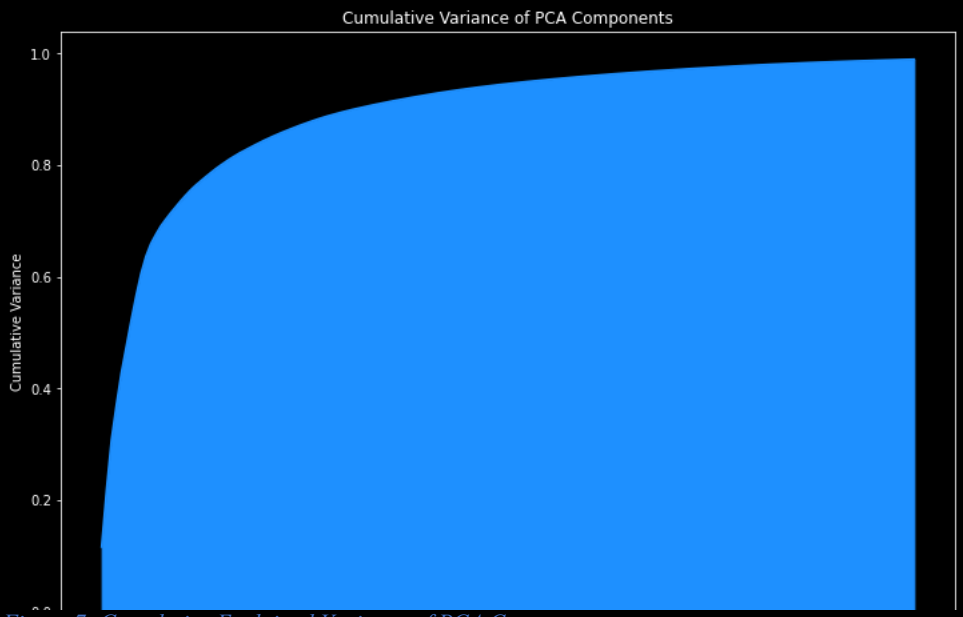

Figure 7: Cumulative Explained Variance of PCA Components

	n Features	Accuracy	Precision	Recall	False Pos	False Neg	AUC
Base Model	300	0.63	0.16	0.53	3,102	972	0.64
Best Accuracy	8	0.88	0.29	0.05	147	1,075	0.53
Best Precision	100	0.66	0.50	0.17	2,776	561	0.63
Best Recall	1	0.13	0.12	1.0	8,652	0	0.51
Best Accuracy PCA	10	0.65	0.16	0.49	2,902	574	0.62
Best Precision PCA	290	0.64	0.16	0.54	3,097	523	0.64
Best Recall PCA	170	0.64	0.17	0.55	3,110	509	0.64

Table 2: Feature Selection Results

Model Tuning

Having determined 100 features to yield optimal results, I then proceeded with hyperparameter tuning. Because many logistic regression hyperparameter options are determined by the solver used, I did a grid search assessing the performance of the following solvers: LBFGS, Liblinear, Newton-CG, SAG, and SAGA. The best performance was achieved using a Newton-CG solver.

Using this solver, I conducted another grid search to determine the best regularization method and strength in addition to if the model should fit an intercept. The Newton-CG does not support L1 regularization, so the options were L2 regularization or none and the strength of regularization, C, was chosen from 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. The grid search found no regularization to be optimal (and therefore C of 0). This resulted in a marginal improvement on the validation set in terms of reducing false negatives and false positives (table 3).

	Accuracy	Precision	Recall	False Pos	False Neg	True Pos	AUC
Base Model	0.63	0.16	0.53	3,102	972	610	0.64
n Feature-Optimized Model	0.66	0.17	0.50	2,776	561	572	0.63
Hyperparameter-Tuned Model	0.67	0.17	0.53	2,756	512	571	0.63

Table 3: Final Model Comparison

Results and Discussion

The final model was then used to predict readmissions on the test set. This resulted in an accuracy of 0.67, precision of 0.17, recall of 0.52, and AUC score of 0.64 (fig. 8, table 4).

Obs \ Pred	Not Readmitted	Readmitted
Readmitted	6,114	2,756
Not Readmitted	512	566

Table 4: Final Model Confusion Table

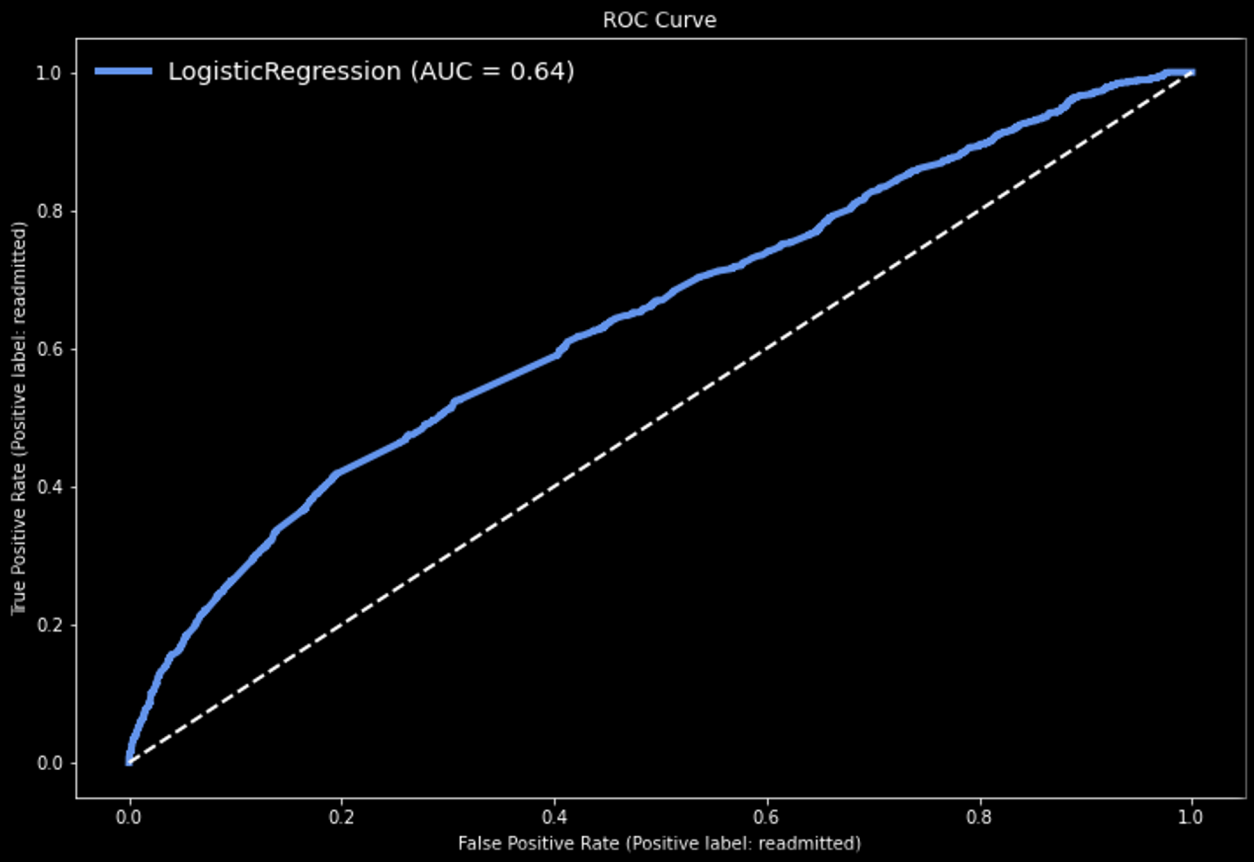

Figure 8: Final Model ROC Curve

Despite optimization, the model still misses a little under half of those that ended up being readmitted within 30 days of discharge. It is possible that another classification algorithm might perform better – both the random forest and neural net models performed well on the training data but suffered from overfitting. Addressing overfitting in either of these models might result in better performance. Additional features may be beneficial as well. For example, the dataset was missing from the housing status of the patient, which might be a strong predictor – those experiencing homelessness face many barriers in managing chronic health conditions resulting in disproportionate use of secondary and tertiary care. Additionally, the inclusion of non-diabetic patients can broaden the application of this model.

Using a logistic regression allows us to compare the relative importance of features in predicting likeliness of readmission by looking at the magnitude of the model coefficients (fig. 9). This can help us understand the risk factors that contribute to readmission. For example, patients discharged to a psychiatric facility have a higher risk of being readmitted. Of note, the features with the largest negative predictive value were if the patient expired or discharged to hospice – once a patient has died, they are certainly not being readmitted. The coefficients can also be contrasted against the most important features as determined by the random forest

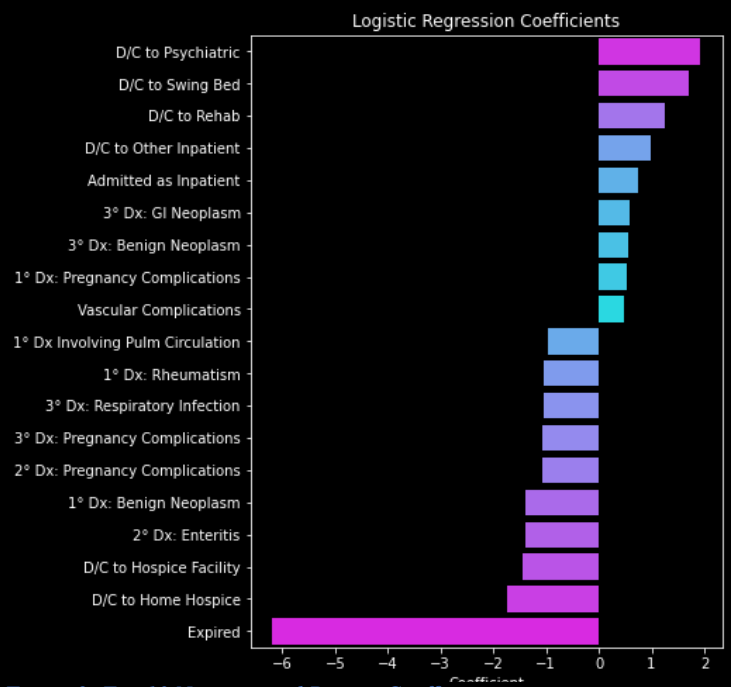

Figure 9: Top 10 Negative and Positve Coefficients

model of which there was little overlap amongst the top features (fig. 10). The feature found to be most important by the random forest was the number of inpatient visits over the past year, followed by weight and age.

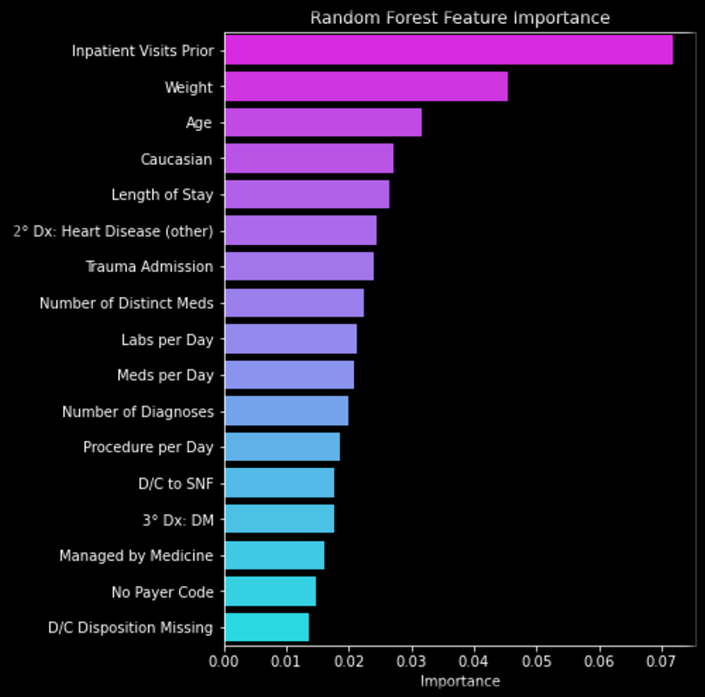

Figure 10: Random Forest Feature Importance

While these results are reported as a binary yes/no, I see using the probability outputs of models as being more informative in a healthcare setting. Providers can use these probabilities as a Likelihood of Readmission Score when determining if a patient is discharged. For example, a classifier would label a patient with a 0.49 chance of readmission as 'not readmitted.' On the other hand, a provider looking at the percentage score would recognize that this patient still has sizeable risk of readmission and might give them reservations over discharging the patient.

Furthermore, providers can run scenarios to see what might reduce the chance of readmission. Providers can look at how different discharge dispositions might impact Likeliness of Readmission – a patient discharged directly to home might have a score of 0.6 but discharging the patient to a SNF might decrease that score. Additionally, other risk factors for readmission can be assessed as well, helping providers ensure that a patient being discharged has the best of chances of staying out of the hospital.

Data Source

Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records," BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.

Appendix A: Model Selection Results

KNN

Training:

Predicted	not readmitted	readmitted
Observed		
not readmitted	40059	30565
readmitted	10	70614

Accuracy:
0.7835367580425917
Recall:
0.9998584050747621
Precision:
0.6979116219768925

Validate:

Predicted	not readmitted	readmitted
Observed		
not readmitted	3782	5033
readmitted	365	768

Accuracy:
0.4573783675110575
Recall:
0.677846425419241
Precision:
0.1323909670746423

Initial Logistic Regression

Training:

Predicted	not readmitted	readmitted
Observed		
not readmitted	46311	24313
readmitted	25367	45257

Accuracy:
0.6482782057091074
Precision:
0.6408161531490711
Recall:
0.6505246514302142

Validate:

Predicted	not readmitted	readmitted
Observed		
not readmitted	5713	3102
readmitted	523	610

Accuracy:
0.6356051467631685
Precision:
0.5383936451897617
Recall:
0.16433189655172414

Naïve Bayes

Training:

Predicted	not readmitted	readmitted
Observed		
not readmitted	8919	61705
readmitted	2423	68201

Accuracy:
0.5459900317172632
Recall:
0.9656915496148618
Precision:
0.5250026942558466

Validate:

Predicted	not readmitted	readmitted
Observed		
not readmitted	1079	7736
readmitted	99	1034

Accuracy:
0.21240450341777242
Recall:
0.912621359223301
Precision:
0.11790193842645381

Neural Net

Training:

Predicted	not readmitted	readmitted
Observed		
not readmitted	70560	64
readmitted	1697	68927

Accuracy:
0.9875325668328047
Recall:
0.9759713411871318
Precision:
0.9990723427693468

Validate:

Predicted	not readmitted	readmitted
Observed		
not readmitted	8069	746
readmitted	972	161

Accuracy:
0.8273019702452754
Recall:
0.14210061782877317
Precision:
0.17750826901874311

Random Forest

```
Training:
Predicted       not readmitted  readmitted
Observed
not readmitted           70624           0
readmitted                   0       70624

Accuracy:
       1.0
Recall:
       1.0
Precision:
       1.0

Validate:
Predicted       not readmitted  readmitted
Observed
not readmitted            7992         823
readmitted                 947         186

Accuracy:
       0.8220747889022919
Recall:
       0.1641659311562224
Precision:
       0.18434093161546086
```

Appendix B: Forward Feature Selection Results

Best Accuracy

```
Training:
Predicted       not readmitted  readmitted
Observed
not readmitted           69375        1249
readmitted               65336        5288

Accuracy:
      0.5285950951517897
Precision:
      0.808933761664372
Recall:
      0.07487539646579067

Validate:
Predicted       not readmitted  readmitted
Observed
not readmitted            8668         147
readmitted                1075          58

Accuracy:
      0.8771612384398875
Precision:
      0.28292682926829266
Recall:
      0.051191526919682
```

Best Precision

```
Training:
Predicted       not readmitted  readmitted
Observed
not readmitted           48354       22270
readmitted               30611       40013

Accuracy:
      0.6256159379247848
Precision:
      0.642438546633913
Recall:
      0.5665637743543271

Validate:
Predicted       not readmitted  readmitted
Observed
not readmitted            6039        2776
readmitted                 561         572

Accuracy:
      0.6645556895858464
Precision:
      0.17084826762246116
Recall:
      0.504854368932038826
```

Best Recall

```
Training:
Predicted       not readmitted  readmitted
Observed
not readmitted            1289       69335
readmitted                   0       70624

Accuracy:
      0.5091257929315813
Precision:
      0.5046049200122893
Recall:
      1.0


Validate:
Predicted       not readmitted  readmitted
Observed
not readmitted             163        8652
readmitted                   0        1133

Accuracy:
      0.13027744270205066
Precision:
      0.11578947368421053
Recall:
      1.0
```

Forward Feature Selection Program:

```
def evaluate(y,y_pred,):
    return m.accuracy_score(y,y_pred),\
           m.precision_score(y,y_pred,pos_label='readmitted'),\
           m.recall_score(y,y_pred,pos_label='readmitted')
coefficients = pd.DataFrame({'features':X_train.columns,
                             'coefficients':abs logistic_regression.coef_[0])}
                           ).sort_values(by = 'coefficients',
                                         ascending = False,
                                         key = lambda x: abs(x)
                                         )
features = []
Results  = pd.DataFrame(columns = ['accuracy','precision','recall'],
                        index   = coefficients.feature)
for f in coefficients.features:
    features.append(f)
    results.loc[f] = evaluate y_validate,LogisticRegression().\
                                        fit(X_train[features],y_train).\
                                        predict(X_validate[features]))
```

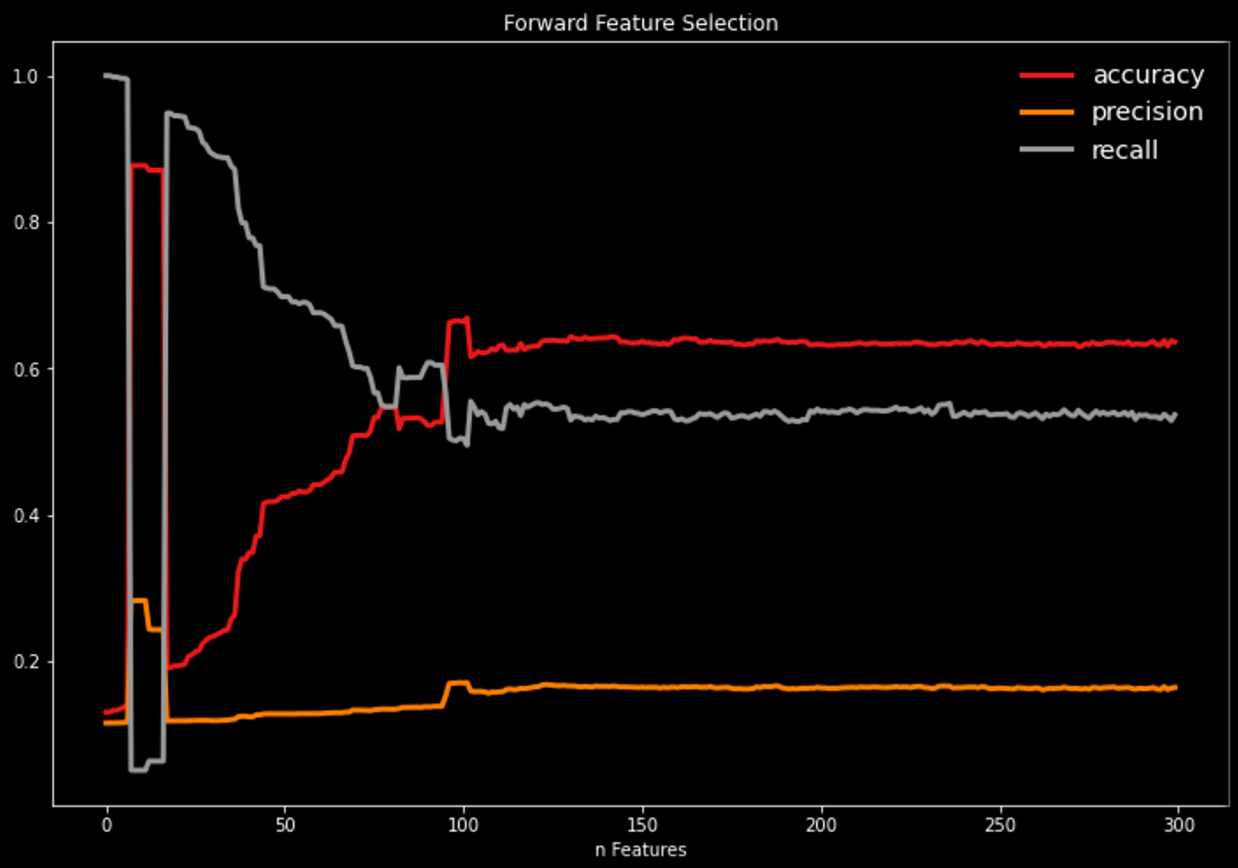

Figure 1: Forward Feature Selection Accuracy, Precion Recal

Appendix C: Forward PCA Selection Results

Best Accuracy

Training:

Training:

Predicted	not readmitted	readmitted
Observed		
not readmitted	47539	23085
readmitted	34196	36428

Accuracy:
0.5944650543724513
Precision:
0.612101557642868
Recall:
0.5158019936565473

Validate:

Predicted	not readmitted	readmitted
Observed		
not readmitted	5913	2902
readmitted	574	559

Accuracy:
0.650583031765179
Precision:
0.16151401329095638
Recall:
0.49338040600176525

Best Precision

Training:

Predicted	not readmitted	readmitted
Observed		
not readmitted	48354	22270
readmitted	30611	40013

Accuracy:
0.6256159379247848
Precision:
0.642438546633913
Recall:
0.5665637743543271

Validate:

Predicted	not readmitted	readmitted
Observed		
not readmitted	6039	2776
readmitted	561	572

Accuracy:
0.6645556895858464
Precision:
0.17084826762246116
Recall:
0.50485436893203882

Best Recall

```
Training:
Predicted       not readmitted  readmitted
Observed
not readmitted           46333       24291
readmitted               26309       44315


Accuracy:
       0.641764839148165
Precision:
       0.6459347578928957
Recall:
       0.6274779111916629


Validate:
Predicted       not readmitted  readmitted
Observed
not readmitted            5705        3110
readmitted                 509         624

Accuracy:
       0.6362082830719743
Precision:
       0.16711301553294056
Recall:
       0.5507502206531333
```

PCA Forward n Components Program:

```
results = pd.DataFrame columns = ['accuracy','precision','recall'],
                         index   = [n for n in range(10,291,10)])
for n in range 10 291 10

    pca = PCA(n_components = n,
              random_state = 1984
    X_train_pca    = pca.fit_transform(X_train)
    X_validate_pca = pca.transform(X_validate)

       results.loc[n] = evaluate(y_validate, LogisticRegression().\
                                             fit(X_train_pca,y_train).\
                                             predict(X_validate_pca))
```

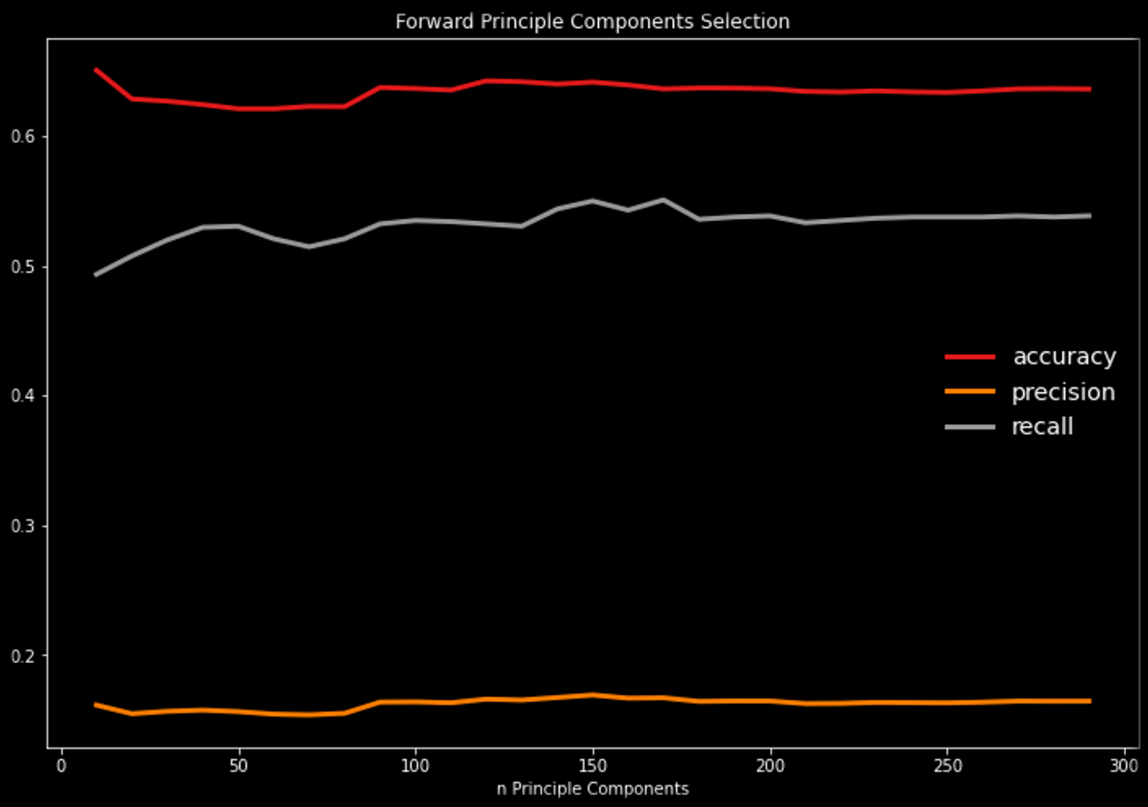

Figure 2: PCA n Components Accuracy, Precision, Recall