Eric Yarger

Classification Analysis

Question Proposal

The question this research paper proposes to answer is:

Is it possible to predict patients who are readmitted using k-nearest neighbor classification from the medical_clean data set, so that our medical organization can take the appropriate steps to minimize preventable readmissions?

Excess readmission rates can cause penalties and fines to be imposed. It is critical for our organization to have the tools and procedures in place for identifying any patient data that can help predict readmission.

Goal

Accurate predictions of the probability that a patient will be readmitted can help inform our executives to make decisions related to patient administration, business decisions, and operational management of our hospitals. The goal of this analysis is to provide a model made up from features in the medical_clean data set to predict patients with a high chance of readmission so that appropriate actions can be taken to minimize unnecessary readmissions.

Classification Method

To quote from Toward Data Science's M. Miller, "Data science or applied statistics courses typically start with linear models, but in its way, K-nearest neighbors is probably the simplest widely used model conceptually. KNN models are really just technical implementations of a common intuition, that things that share similar features tend to be, well, similar. This is hardly a deep insight, yet these practical implementations can be extremely powerful, and, crucially for someone approaching an unknown dataset, can handle non-linearities without any complicated data-engineering or model set up." (Miller, 2019).

KNN classifies on the idea that every data point near or next to each other is classified in the same class. KNN classifies new data points in the selected data set based on similarity. The 'NN' in KNN stands for Nearest Neighbor, which speaks to the classification method. The 'K' in KNN represents the number of neighbors specified to perform the classification. This is set by the programmer before KNN is ran.

Expected outcomes of KNN classification for this analysis are:

- There will be a small number of variables from the chosen data set that will provide the majority of the models' predictive capacity in determining readmission rates

- The model will provide a tool that predicts readmission rates from variables in the chosen data set with a high level of accuracy. This will be validated through classification reports and evaluation metrics to ensure accuracy.

Method Assumption

An assumption of KNN is that it assumes similar data points exist near each other. The value of each data point is determined by the data points around it in KNN. The classification of each data point is determined by the number (K) of points checked around it.

To build on this assumption with another assumption of KNN is that a suitable K value is chosen to create an accurate KNN classification model. If the K value selected is too low the model can suffer from being too specific, overfitting the data set. If the K value selected is too high, the model can suffer from being too general in its classifications, ultimately resulting in failure to provide meaningful predictions, underfitting the data set (Yildirim, 2020).

According to Medium's writer username Vishalmendekarhere, KNN also assumes that (Vishalmendekarhere, 2021):

- Data is in feature space, and can be measured by distance metrics
- Each training data point consists of a set of vectors and class labels associated with that vector.
- 'K' should have an odd number in the case of 2 class classification.

Packages and libraries used

Packages and Libraries Used in this analysis	Justification for how Package or Library supports the analysis
matplotlib.pyplot	Provides an implicit way of plotting data for visualizations and analysis.
pandas	Feature rich data analysis and manipulation tool for Python. Used throughout the analysis.
numpy	Library for Python that adds support for large, multidimensional arrays.
seaborn	Visualization and graph creation tool for Python.
missingno	Library that provides graphs for visualizing missing data in a data set.

scipy	Library for scientific and technical computing in Python.
Scipy.stats import zscore	Used for calculating zscores.
Sklearn.feature_selection import SelectKBest, chi2	SelectKBest used for selecting the features with the highest k score. Chi2 used for computing chi2 stats.
Sklearn.neighbors import KNeighborsClassifier	Classifier implementing k-nearest neighbors vote. Used for KNN classification.
sklearn.metrics import accuracy_score	Used for computing subset accuracy from results.
Sklearn.model_selection import cross_val_score, train_test_split	Cross_val_score evaluates score by cross validation. Train_test_split splits the data into respective training and testing data sets.
Sklearn.metrics import classification_report	Classification_report computes and presents a report showing the main metrics from classification
Sklearn.model_selection import GridSearchCV	Implements a fit and score method. The parameters of the estimator used is optimized by cross-validated-grid-search.
Sklearn import confusion_matrix	Computes a confusion matrix - helps evaluating and visualizing the accuracy of our classification.
Sklearn.metrics import roc_auc_score	Computes ROC and AUC from our prediction scores.
Sklearn import metrics	This module includes score function and performance methods useful in our analysis.

Preprocessing Data

The main goal for data preprocessing that is relevant to our research question is to ensure that the data set chosen is optimally prepared for the KNN model. This ensures that the results from the model are as accurate as possible. This goal includes:

- Removal or imputation of missing data and outliers.
- Creation of dummy variables for any categorical variables.
- Scaling the chosen data set. Where KNN takes k-number of samples closest to the new point to predict, features need to be the same scale to output accurate results.

By ensuring our data preparation meets these parameters we will have an optimal starting point for answering the question posed in section A1.

Variables Chosen for Analysis

Listed below are all of the initial data set variables used for classification analysis. The features were selected in Step 9 of the data preparation stage using SelectKBest(). The top 10 independent features from the data set were selected to perform this analysis. Data type for categorical features was optimized to int8 in step 12 of the data preparation.

TARGET VARIABLE	**CATEGORICAL OR CONTINUOUS**	**DATA TYPE**
ReAdmis	Categorical	int8
PREDICTOR VARIABLES	**CATEGORICAL OR CONTINUOUS**	**DATA TYPE**
Initial_days	Continuous	float64
TotalCharge	Continuous	float64
Children_1	Categorical	int8
Children_4	Categorical	int8
Children_6	Categorical	int8
Marital_Divorced	Categorical	int8
vitD_supp_1	Categorical	int8
vitD_supp_2	Categorical	int8
Services_CT_Scan	Categorical	int8
Services_Intravenous	Categorical	int8

Steps for Analysis

Listed and explained below are the steps to prepare the data for analysis.

Step	Action and explanation
1	Load the data and initial visualization to see shape/size/type of data features.
2	Rename survey features to provide a more usable / readable data set. Plot Pairplots X-Axis is ReAdmis, Y-axis is independent variables
3	Address missing data, duplicates, and outliers. This process makes the data more applicable for KNN classification. ReAdmis replace Yes/No with 1/0 for utility.

4	Look at the correlation between variables using correlation table and heatmaps. This provides visual and statistically useful metrics for feature selection.
5	Create dummy variables - makes categorical features applicable for KNN. Rename any applicable variables.
6	Drop demographic features that won't be used in the analysis. These features have little correlation, many different values, and are not statistically useful for the purpose of the analysis.
7	Create variables y = ReAdmis, X = df with ReAdmis dropped. This separates the target and predictor features.
8	Min/Max Scale features. Scaling is necessary to give the same value limits to the features, float from 0 to 1. This makes the features more useful for KNN
9	Implement SelectKBest to identify features for selection. This selects n (in this case 10) best features according to the k highest score. Provides us with the best features for KNN. Reference: (Sklearn.feature_selection.SelectKBest, N.d).
10	Rejoin variable 'y' and selected feature variable 'dfs' for fully prepared dataset.
11	Graphs and Summary Statistics for visualizing data stats and shapes.
12	.replace() and .astype = int8 for categorical variables. Convert categorical variables into minimal datatype for calculations.
13	Assign prepared features to y = ReAdmis, X = Prepared independent features. Prepares the data for KNN and Cross validation
14	Import necessary libraries for KNN classification. This prepares the environment to have the necessary tools for KNN. Split data into Training (X_train, y_train) and Testing (X_test, y_test). This allows us to have (in the case of this analysis) 70% of the data for training and 30% for testing.

Code input and output for each step in preprocessing the data set is labeled and identified in the supplemental copy of my Jupyter Notebook used for the analysis under the section 'Data Preparation'.

Intermediate Calculations and Output

The analysis technique used was KNeighborsClassifier() for classification to predict readmission. The N-Neighbors number best used for this model was calculated using KneighborsClassifier with all default settings with the exception of n-neighbors. The optimal n-neighbors was calculated using GridSearchCV, with KNeighborsClassifier set to all default parameters, cv=5 and a parameter grid dictionary from 1 to 50. Printing the result of GridSearchCV yielded that 4 is the best n-neighbor for our classification model.

Next, the KNeighborsClassifier is assigned to the variable 'knc'. The model is fitted using .fit() to the training X and y data sets. Next KneighborsClassifier is computed again using 4 for n_neighbors and all other parameters set to their default. I called .predict() on the model variable, which uses the learned parameters from .fit() to perform predictions on new unseen data points, the y_pred data set. Discussion on .fit() and .predict() methods from an article on Towards Data Science (Myrianthous, 2021). Next, the model's accuracy score, classification report, and confusion matrix are calculated. Model accuracy is .96705. Precision is .97(0), .96(1). These figures indicate a highly accurate model for classification of readmission. Below is the screenshot of the input of the intermediate calculations and their outputs. This includes setting up the KNeighborsClassifier, the model's Accuracy Score, Classification Report, and Confusion Matrix.

[ ]:

09]:
```
# Import GridSearchCV for cross validation of model
# Code reference (Okamura, 2020).

paramgrid = {'n_neighbors': np.arange(1, 50)}
knc = KNeighborsClassifier()
knccv = GridSearchCV(knc , paramgrid, cv=5)

# Fit the model to training data.
knccv.fit(X_train, y_train)

print('The best n_neighbors for the model: {}'.format(knccv.best_params_))
```

```
The best n_neighbors for the model: {'n_neighbors': 4}
```

10]:
```
# Print ot the best score for classification model
print('The best classification score for the model: {:.6f}'.format(knccv.best_score_))
```

```
The best classification score for the model: 0.970981
```

[ ]:

Section D: Data Analysis

11]:
```
# KneighborsClassifier code method reference (Klein, 2022)

knc = KNeighborsClassifier(n_neighbors=4)

knc.fit(X_train,y_train)
```

11]:
```
KNeighborsClassifier(n_neighbors=4)
```

12]:
```
y_pred = knc.predict(X_test)
```

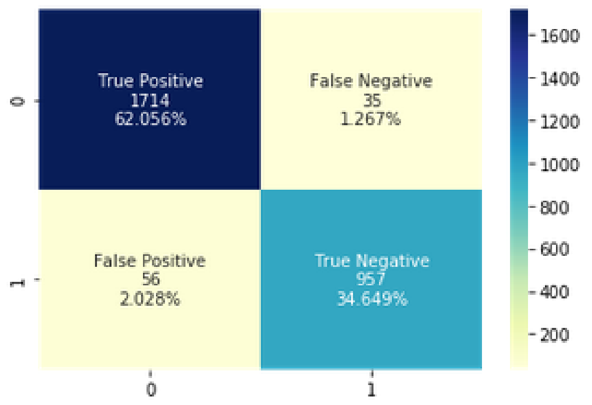
13]:
```
print('KNN model accuracy:', accuracy_score(y_test, y_pred))
```

```
KNN model accuracy: 0.9670528602461984
```

14]:
```
print( classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      1749
           1       0.96      0.94      0.95      1013

    accuracy                           0.97      2762
   macro avg       0.97      0.96      0.96      2762
weighted avg       0.97      0.97      0.97      2762
```

```
15]:
     confmatrix = confusion_matrix(y_test, y_pred)
     print(confmatrix)

     [[1714   35]
      [  56  957]]

16]: # Confusion matrix visualization reference (Aruchamy, 2021).

     matnames = ['True Positive', 'False Negative', 'False Positive', 'True Negative']
     matcounts = ["{0:0.0f}".format(value) for value in confmatrix.flatten()]
     matpercent = ["{0:.3%}".format(value) for value in confmatrix.flatten()/np.sum(confmatrix)]
     labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(matnames, matcounts, matpercent)]
     labels = np.asarray(labels).reshape(2,2)
     sns.heatmap(confmatrix, annot=labels, fmt='', cmap='YlGnBu')

16]: <matplotlib.axes._subplots.AxesSubplot at 0x1b9154fb1c8>
```

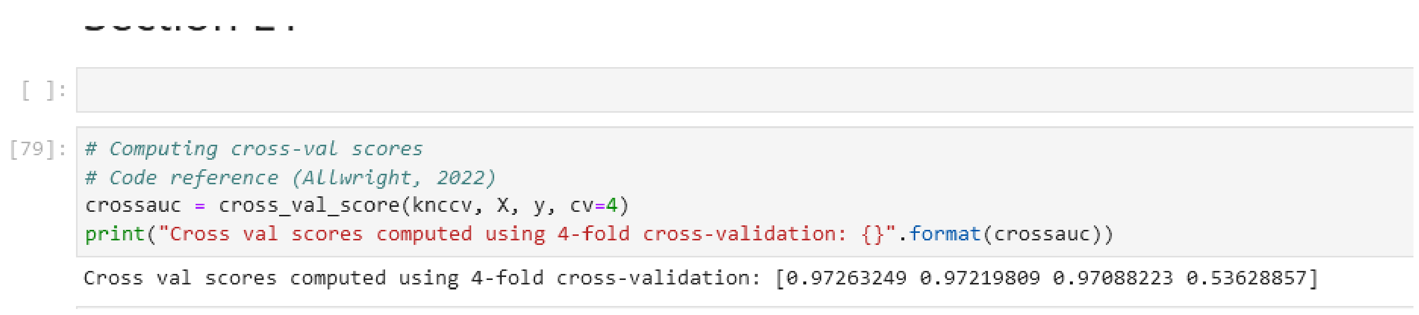


Below is a discussion on the parameters selected in this analysis. Parameter descriptions gathered from scikit learn's KNeighborsClassifier documentation (Sklearn.neighbors.KNeighborsClassifier, n.d.).

- N-neighbors: set to 4
  - This is the parameter to set the number of neighbors to use for KNeighbor queries. For example, our number for this analysis is initially set to 4. This means that the 4 nearest data points to the point being classified will be used for the classification decision. The optimal n-neighbor will be determined later in the analysis using GridSearchCV.
- Weights - default
  - The default setting was used, which is 'uniform'. All points are equally weighted.

- Algorithm - default
    - The default setting was used, which is 'auto'. Auto allows KNeighborsClassifier to select the best algorithm based on the parameters passed to the method.
- Leaf-size - default
    - The default setting was used, which is 30. This parameter affects the speed and construction of a query.
- P - default
    - The default setting was used, which is 2. This specifies that KNeighborsClassifier uses euclidian_distance. Euclidean distance is a widely used mathematical formula for calculating distance in metric space.
- Metric - default
    - The default setting was used, which is 'minkowski'. In conjunction with p = 2 from above this metric is equivalent to using the standard Euclidean metric.
- Metric_params - default
    - Default setting 'none' used. This parameter would allow for additional keyword arguments.
- n-Jobs ; default
    - Default setting 'none' used. This parameter specifies the number of parallel jobs to run.

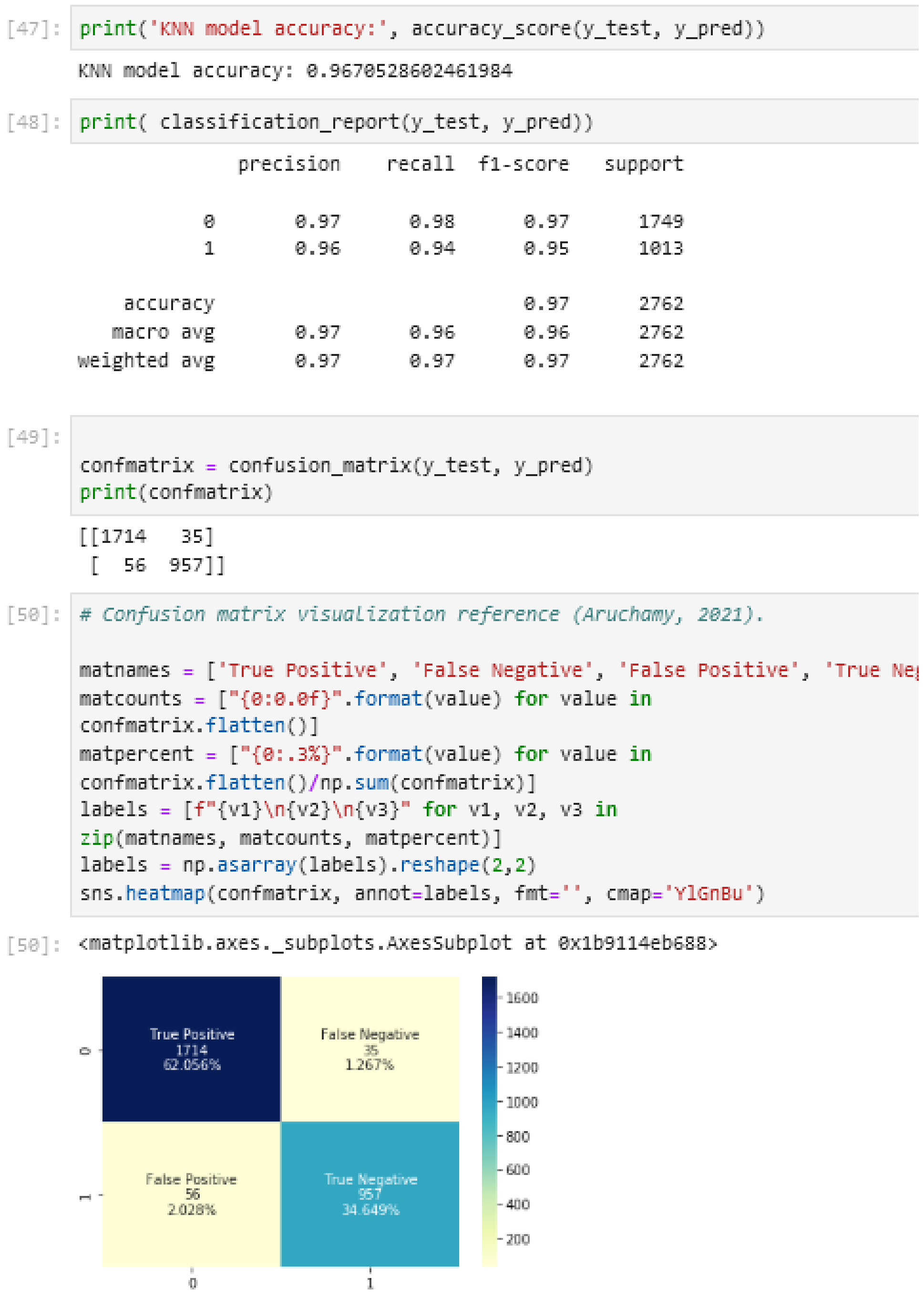
Cross validation of the model was computed using cross_val_score and 4 fold cross validation. This was done as a simple cross validation technique and to assess over-fitting of the model. It also tests whether our model can generalize over the entire dataset (Allwright, 2022). Below is a screenshot of the code used for cross validation.



```
[ ]:

[79]: # Computing cross-val scores
      # Code reference (Allwright, 2022)
      crossauc = cross_val_score(knccv, X, y, cv=4)
      print("Cross val scores computed using 4-fold cross-validation: {}".format(crossauc))

      Cross val scores computed using 4-fold cross-validation: [0.97263249 0.97219809 0.97088223 0.53628857]
```

Accuracy and AUC

The accuracy of the model is the fraction of predictions that our model predicted correctly. The accuracy of the classification model is calculated using sklearn's accuracy_score, a confusion matrix, and looking at precision from the classification report. A screenshot of the code and output is included below.



```
[47]: print('KNN model accuracy:', accuracy_score(y_test, y_pred))

KNN model accuracy: 0.9670528602461984

[48]: print( classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.97      0.98      0.97      1749
           1       0.96      0.94      0.95      1013

    accuracy                           0.97      2762
   macro avg       0.97      0.96      0.96      2762
weighted avg       0.97      0.97      0.97      2762

[49]:
confmatrix = confusion_matrix(y_test, y_pred)
print(confmatrix)

[[1714   35]
 [  56  957]]

[50]: # Confusion matrix visualization reference (Aruchamy, 2021).

matnames = ['True Positive', 'False Negative', 'False Positive', 'True Ne
matcounts = ["{0:0.0f}".format(value) for value in
confmatrix.flatten()]
matpercent = ["{0:.3%}".format(value) for value in
confmatrix.flatten()/np.sum(confmatrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
zip(matnames, matcounts, matpercent)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confmatrix, annot=labels, fmt='', cmap='YlGnBu')

[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1b9114eb688>
```

Accuracy using sklearn's accuracy_score is .96705. This module compares test vs predicted values for ReAdmis. Accuracy_score represents the ratio of True Positives and True Negatives out of all predictions. The 'precision' column from the classification report shows .97 for '0' and .96 for '1'. Precision is the total True

Positive count divided by the sum of True Positives + False Positives. This is included not because accuracy is the same as precision, but is more a validation that our accuracy in predicting ReAdmis is correct. The confusion matrix looks at the model's performance and accuracy for predictions vs actual data. It's useful in visualizing the model's performance. The sum of True Positive and True Negative from the confusion matrix is .96705, the same as our accuracy score.

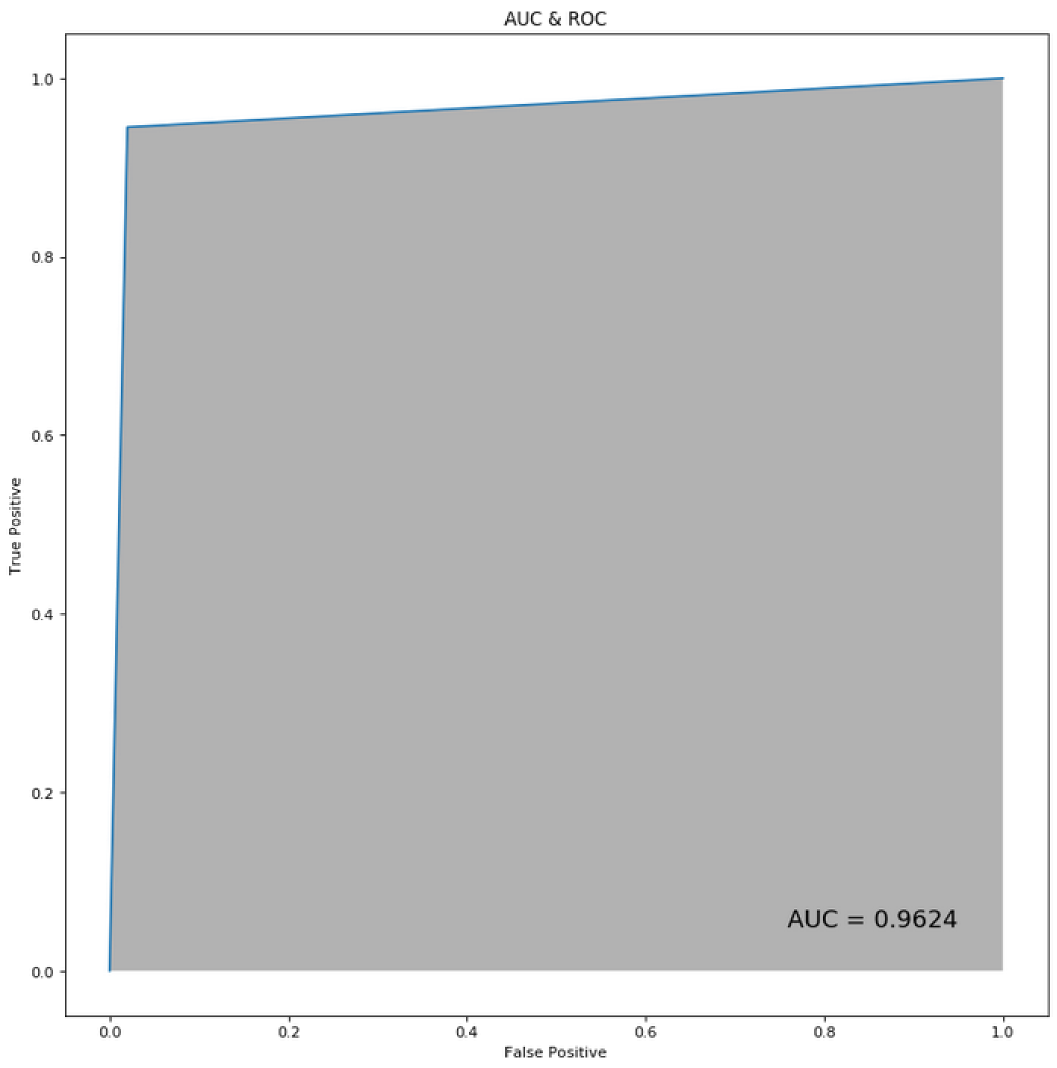
With a model accuracy of ~.97 the model predicts ReAdmis output accurately 97% of the time. This is a robust accuracy metric, but it isn't perfect. False Positive, where a patient was not readmitted but the model predicted they were, and False Negative, where a patient was readmitted but the model predicted they weren't, stand at a cumulative ~3%.

The AUC graph for the classification model is shown below. All code used to generate the AUC graph is included in the supplemental pdf of my Jupyter Notebook under section E1.

```
[79]: # Computing cross-val scores
      # Code reference (Allwright, 2022)
      crossauc = cross_val_score(knccv, X, y, cv=4)
      print("Cross val scores computed using 4-fold cross-validation: {}".format(crossauc))
```

```
Cross val scores computed using 4-fold cross-validation: [0.97263249 0.97219809 0.97088223 0.53628857]
```

```
[78]: # AUC and ROC Curve
      # code reference (Kharwal, 2022)
      auc = metrics.roc_auc_score(y_test, y_pred)

      false_positive_rate, true_positive_rate, thresolds = metrics.roc_curve(y_test, y_pred)


      plt.figure(figsize=(12, 12), dpi=80)
      plt.plot(false_positive_rate, true_positive_rate)
      plt.xlim([0, 1])
      plt.ylim([0, 1])

      plt.axis('scaled')
      plt.fill_between( false_positive_rate,true_positive_rate,facecolor='grey',alpha=0.6)
      plt.text(0.95, 0.05, 'AUC = %0.4f' % auc, ha='right', fontsize=16, weight='normal', color='black')

      plt.title("AUC & ROC")
      plt.xlabel("False Positive")
      plt.ylabel("True Positive")

      plt.show()
```



The AUC (area under the ROC curve) measures the area under the ROC (receiver operating characteristic curve) The ROC shows the performance of our classification model at every classification threshold. It plots True Positive and False Positive rates.

The AUC of the classification model is .9624, calculated using metrics.roc_auc_score. That is to say, the model AUC of .9624 can be said to predict patient readmission accurately 96.24% of the time. AUC is a measure of performance across all classification thresholds. AUC can be interpreted as the odds that our classification model ranks a positive example higher than a random negative example (Classification: ROC curve and AUC, n.d.). The range of AUC is from 0 to 1. A model with an AUC of 0 would predict wrongly 100% of the time. A model with an AUC of 1 would predict accurately 100% of the time.

Implications and Results

The results of the classification analysis is that the model is 96.7% accurate in predicting readmission, using the top 10 predictor variables chosen using SelectKBest. The AUC for the model is 96.24%, indicating that the model accurately predicts between positive and negative sample data. The results from the model accuracy and AUC indicate that the model is statistically and practically significant for use answering the question of this analysis. The model can be used to classify patients who will be readmitted.

The implication of the classification analysis is that the model can be used to predict readmission rates from data gathered by the patient on their initial visit with high accuracy.

Limitations of This Analysis

One limitation of the data analysis lies in the data set's suitableness for KNN classification analysis. When looking at correlation and feasibility of each feature for prediction selection, TotalCharge and Initial_days appeared to be highly correlated with ReAdmis, while all other features were limited in correlation. Correlation doesn't define classification accuracy, but it should be noted that the classification ability of the variables is not distributed evenly at all. To test this I ran the model with Initial_days and TotalCharge removed, and model accuracy was .6004 and AUC was .4964. These numbers are a wild call from the figures with these features included, and indicate that the model is nearly equally likely to classify figures correctly only half the time. When a model is so dependent on two predictive features for the majority of its classification ability, this is a limitation in the overall utility of the model if the data gathered from patients is ever altered.

Recommended Course of Action

The course of action recommended for our organization, in answer to the question posed in A1 from this analysis, is to use this model for effectively classifying patients into (a) patients who are likely to be readmitted, and (b) patients who are not likely to be readmitted. This model provides a simple, accurate, and effective tool for initial classification. It should be said that this model should not be the only tool used by the organization for this action. Accurate readmission rate analysis is highly important for our organization, and this should be used as only one classification tool in a collective toolkit in managing, predicting, and treating readmission rates as a business performance objective.

Third-Party Sources or Code

Bushmanov, Sergey. (2019, January 28). *How to remove Outliers in Python?.* Stackoverflow. https://stackoverflow.com/questions/54398554/how-to-remove-outliers-in-python

Pandas.get_dummies. (N.d.). pandas. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

Aruchamy, Vikram. (2021, September 29). *How to Plot Confusion Matrix in Python And Why You Need To?* Stackvidhya.com. https://www.stackvidhya.com/plot-confusion-matrix-in-python-and-why/

Sklearn.feature_selection.SelectKBest. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

Bprasad26. (2022, March 19). *Feature selection with SelectKBest in Scikit learn*. Retrieved from https://lifewithdata.com/2022/03/19/feature-selection-with-selectkbest-in-scikit-learn/

Zach. (2021, September 9). *How to calculate AUC (Area under curve) in Python*. Retrieved from https://www.statology.org/auc-in-python/

Kharwal, A. (2022, June 3). *AUC and ROC curve using Python*. Retrieved from https://thecleverprogrammer.com/2021/04/07/auc-and-roc-curve-using-python/

Parameter "stratify" from method "train_test_split" (scikit learn). (n.d.). Retrieved from https://stackoverflow.com/questions/34842405/parameter-stratify-from-method-train-test-split-scikit-learn

Allwright, Stephen. (2022, June 27). *Using cross_val_score in sklearn, simply explained*. Retrieved from https://stephenallwright.com/cross_val_score-sklearn/

Okamura, S. (2020, December 30). *GridSearchCV for beginners*. Retrieved from https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee

Klein, Bernd. (2022, July 7.). *9. K-nearest-Neighbor classifier with sklearn*. Retrieved from https://python-course.eu/machine-learning/k-nearest-neighbor-classifier-with-sklearn.php

HSources

Miller, M. (2019, October 18). The basics: KNN for classification and regression. Retrieved from https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955

Yıldırım, S. (2020, December 11). How important is the K in KNN algorithm. Retrieved from https://towardsdatascience.com/how-important-is-the-k-in-knn-algorithm-3b6fce726110

Vishalmendekarhere. (2021, January 22). It's all about assumptions, pros & cons. Retrieved from https://medium.com/swlh/its-all-about-assumptions-pros-cons-497783cfed2d

Sklearn.feature_selection.SelectKBest. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

Classification: ROC curve and AUC. (n.d.). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Sklearn.neighbors.KNeighborsClassifier. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Myrianthous, G. (2021, March 14). Fit() vs predict() vs fit_predict() in Python scikit-learn. Retrieved from https://towardsdatascience.com/fit-vs-predict-vs-fit-predict-in-python-scikit-learn-f15a34a8d39f

Allwright, Stephen. (2022, June 27). *Using cross_val_score in sklearn, simply explained*. Retrieved from https://stephenallwright.com/cross_val_score-sklearn/