Peyton Bailey

October 6th, 2025

D603 Machine Learning – Task 1

B. Describe the purpose of this data mining report by doing the following:

1. Propose one question relevant to a real-world organizational situation that you will answer using one of the following classification methods:

I propose the following question: "Can we predict whether a telecommunications customer will churn in the next month using their service and account features?" I will be using Random Forest Classification to answer this question.

2. Define one goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

The goal is to identify customers at high risk of churning so that targeted retention strategies can be implemented to reduce churn rates and retain profitable customers.

C. Explain the reasons for your chosen classification method from part B1 by doing the following:

1. Explain how the classification method you chose analyzes the selected dataset. Include expected outcomes.

Random forest is a machine learning algorithm popular for handling mixed data types and nonlinear relationships. It uses an ensemble learning method to make predictions by building many individual learning trees and combining their results. It assigns data to categories and classifies them based on a majority vote for classification purposes (Sci-kit developers, 2024).

2. List the packages or libraries you have chosen for Python or R, and justify how *each* item on the list supports the analysis.

Pandas – This package allowed me to import and manipulate the dataset within a data frame. This includes organizing and cleaning the data, dropping unnecessary columns, and handling missing data.

Numpy – This is a fundamental library for numerical and scientific computing, which aided my analysis.

Scikit-learn – This package heavily supports machine learning. It allowed me to utilize the Random Forrest algorithm to fit onto my training data and create a model for classifying the data. It also allowed me to import the train-test feature and split the data into training, validation, and test sets. This package has many features that enable users to analyze metrics such as accuracy and precision, ultimately allowing for hyperparameter tuning to get an optimized model. Furthermore, this package also aided the preprocessing steps by enabling me to encode the categorical variables using label encoders.

Matplotlib – This package supports the creation of visualizations, which I used to view a bar graph of features ranked by their level of importance.

D. Perform data preparation for the chosen dataset by doing the following:

1. Describe one data preprocessing goal relevant to the classification method from part B1.

Data preprocessing aims to ensure all categorical variables are encoded, missing values are imputed, and irrelevant columns are removed, making the dataset compatible with a Random Forest model.

2. Identify the initial dataset variables that you will use to perform the analysis for the classification question from part B1 and classify *each* variable as continuous or categorical.

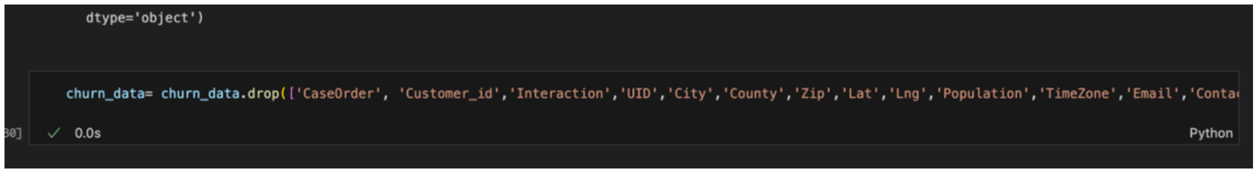
Categorical Variables- State, Area, Marital, Gender, Churn, Contract, Port Modem, Tablet, Internet Service, Phone, Multiple, Online Security, Online Backup, Device Protection, Tech

Support, Streaming TV, Streaming Movies, Paperless Billing, Payment Method, Item 1, Item 2, Item 3, Item 4, Item 5, Item 6, Item 7, Item 8.

Continuous Variables - Age, Income, Outage Sec Per Week, Yearly Equipment Failure, Tenure, Monthly Charge, Bandwidth GB Year.

3. Explain *each* of the steps used to prepare the data for the analysis. Identify the code segment for *each* step.

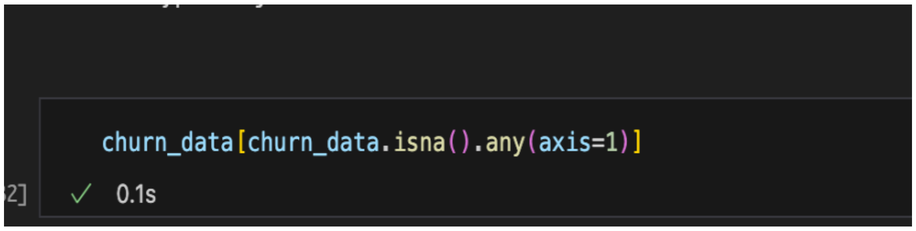
After uploading my dataset into a data frame, the first step was to view the columns and identify any irrelevant columns to the analysis, such as identifiers and redundant variables. Those columns were then dropped.



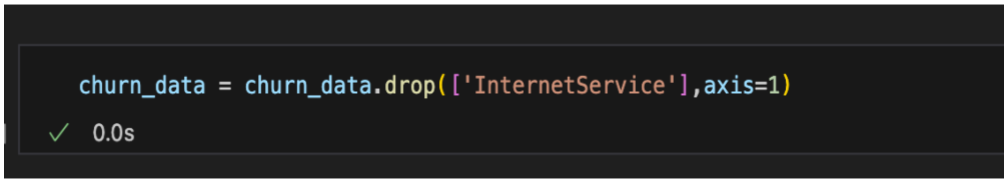
```
    dtype='object')

churn_data= churn_data.drop(['CaseOrder', 'Customer_id','Interaction','UID','City','County','Zip','Lat','Lng','Population','TimeZone','Email','Conta
✓ 0.0s                                                                                                    Python
```

Next, I handled missing data. I did this by first determining which columns contained missing data. I determined that the only column missing data was the Internet Service category, which had roughly 2100 missing instances. I decided to drop that column altogether, which left me with a complete dataset with no missing values.

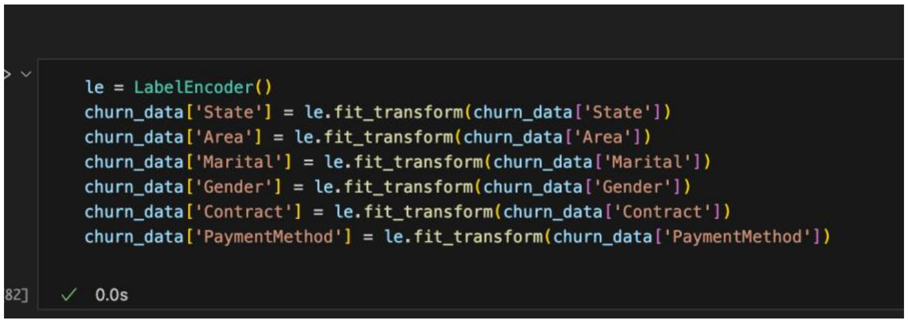


```
churn_data[churn_data.isna().any(axis=1)]
✓ 0.1s
```



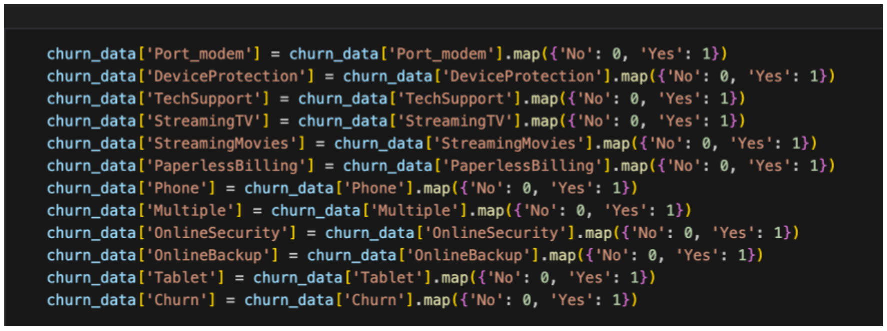
```
churn_data = churn_data.drop(['InternetService'],axis=1)
✓ 0.0s
```

The final step in the preprocessing phase was to encode the categorical variables. I used the label encoder for the State, Area, Marital, Gender, Contract, and Payment Method categories. For the binary columns, which only had "Yes" or "No" as possible outcomes, I directly mapped "No" to 0 and "Yes" to 1 as indicators.



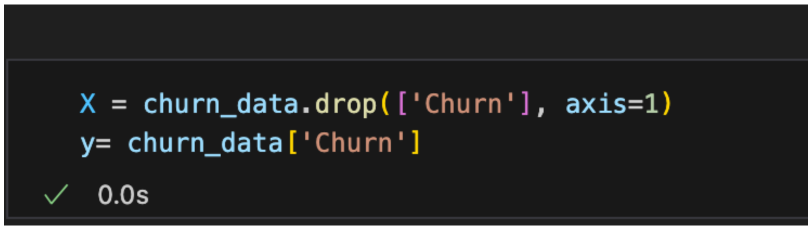
```
le = LabelEncoder()
churn_data['State'] = le.fit_transform(churn_data['State'])
churn_data['Area'] = le.fit_transform(churn_data['Area'])
churn_data['Marital'] = le.fit_transform(churn_data['Marital'])
churn_data['Gender'] = le.fit_transform(churn_data['Gender'])
churn_data['Contract'] = le.fit_transform(churn_data['Contract'])
churn_data['PaymentMethod'] = le.fit_transform(churn_data['PaymentMethod'])
```

82] ✓ 0.0s



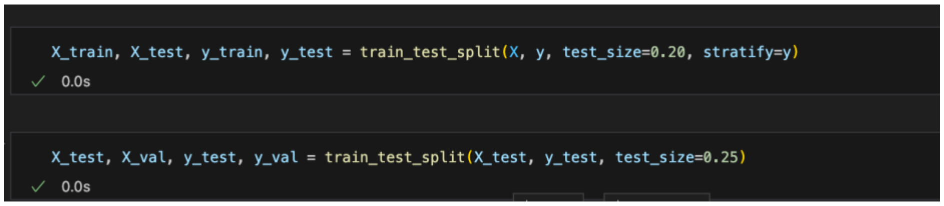
```
churn_data['Port_modem'] = churn_data['Port_modem'].map({'No': 0, 'Yes': 1})
churn_data['DeviceProtection'] = churn_data['DeviceProtection'].map({'No': 0, 'Yes': 1})
churn_data['TechSupport'] = churn_data['TechSupport'].map({'No': 0, 'Yes': 1})
churn_data['StreamingTV'] = churn_data['StreamingTV'].map({'No': 0, 'Yes': 1})
churn_data['StreamingMovies'] = churn_data['StreamingMovies'].map({'No': 0, 'Yes': 1})
churn_data['PaperlessBilling'] = churn_data['PaperlessBilling'].map({'No': 0, 'Yes': 1})
churn_data['Phone'] = churn_data['Phone'].map({'No': 0, 'Yes': 1})
churn_data['Multiple'] = churn_data['Multiple'].map({'No': 0, 'Yes': 1})
churn_data['OnlineSecurity'] = churn_data['OnlineSecurity'].map({'No': 0, 'Yes': 1})
churn_data['OnlineBackup'] = churn_data['OnlineBackup'].map({'No': 0, 'Yes': 1})
churn_data['Tablet'] = churn_data['Tablet'].map({'No': 0, 'Yes': 1})
churn_data['Churn'] = churn_data['Churn'].map({'No': 0, 'Yes': 1})
```

The data was then split into X and y. The X dataset was my predictor variables, which included every column except for Churn, which was my response variable that I assigned to y. Finally, the data was split into training, validation, and test sets.



```
X = churn_data.drop(['Churn'], axis=1)
y= churn_data['Churn']
```

✓ 0.0s



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, stratify=y)
```

✓ 0.0s

```
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.25)
```

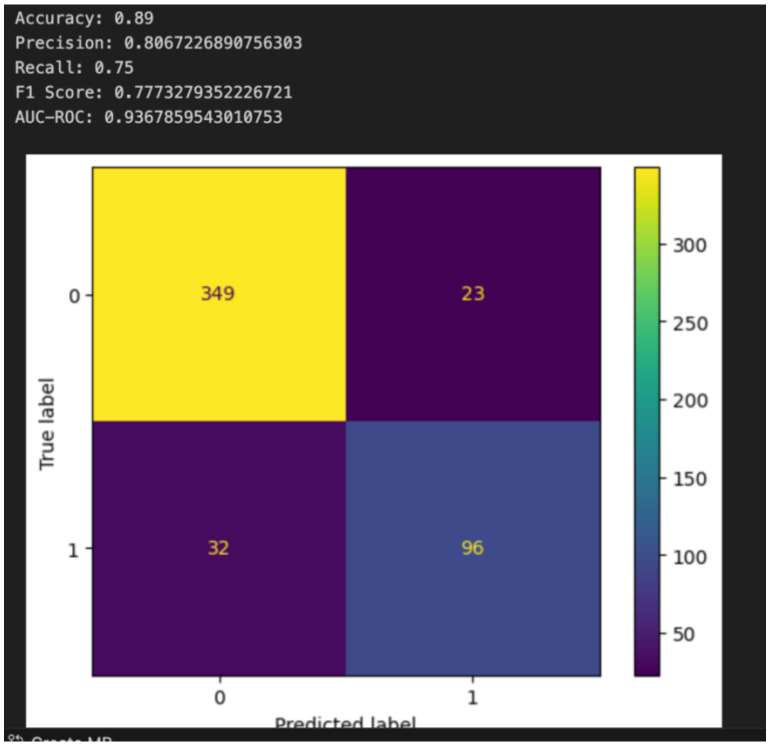
✓ 0.0s

4. Provide a copy of the cleaned dataset.

E. Perform the data analysis and report on the results by doing the following:

1. Split the data into training, validation, and test datasets and provide the file(s).

2. Create an initial model using the training dataset and provide a screenshot of the following metrics:



```
Accuracy: 0.89
Precision: 0.8067226890756303
Recall: 0.75
F1 Score: 0.7773279352226721
AUC-ROC: 0.9367859543010753
```

- **accuracy**
- **precision**
- **recall**
- **F1 score**
- **AUC-ROC**
- **confusion matrix**

3. Perform hyperparameter tuning on the validation dataset using k-fold cross-validation to find the optimized model. Provide the following in the submission:

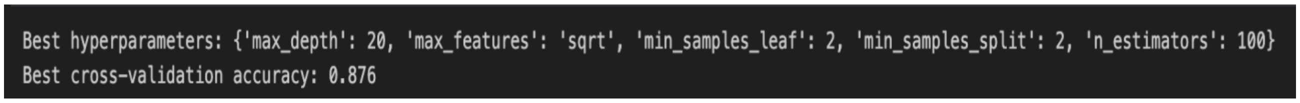
- **identification of which hyperparameters were selected for tuning**

The chosen hyperparameters were the number of estimators, maximum depth, minimum samples per split, minimum samples per leaf, and maximum features.

- **justification of the selection of these hyperparameters**

The number of estimators is the number of trees used to fit the model. A higher number of trees generally improves accuracy but increases computation (DataCamp, 2024). The goal is to find a balance that optimizes predictive performance and runtime. The maximum depth controls the maximum depth of each tree. A shallow tree may underfit while a deep tree may overfit. Therefore, choosing the correct value for this hyperparameter was also very important. Minimum samples per split and leaf reduce the risk of splitting on minimal and noisy data subsets. Finally, the maximum features hyperparameter limits the number of features checked at each split, introducing more randomness to the ensemble.

- **screenshot of the best hyperparameters**

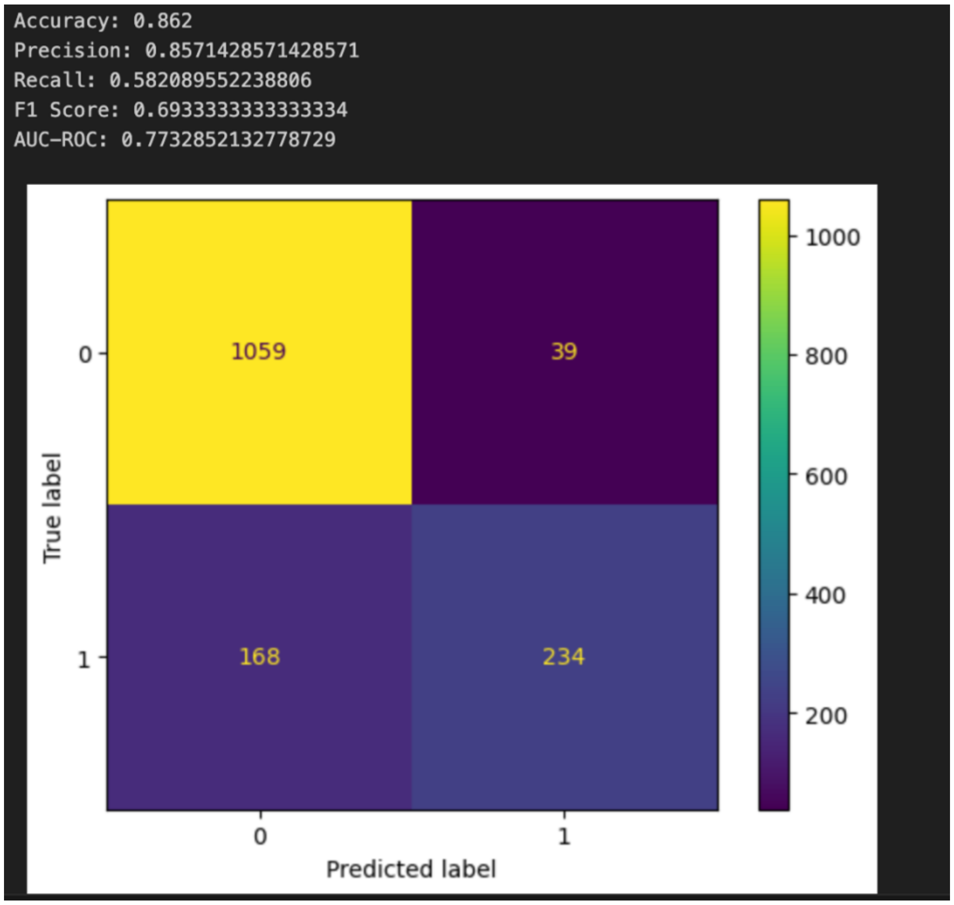


```
Best hyperparameters: {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
Best cross-validation accuracy: 0.876
```

(Scikit-learn developers, 2024).

4. Use the optimized model identified in part E3 to make predictions using the test dataset and provide a screenshot of the following metrics:

- **accuracy**
- **precision**
- **recall**
- **F1 score**
- **AUC-ROC**
- **confusion matrix**



```
Accuracy: 0.862
Precision: 0.8571428571428571
Recall: 0.582089552238806
F1 Score: 0.6933333333333334
AUC-ROC: 0.7732852132778729
```

F. Summarize your data analysis by doing the following:

1. Compare and discuss the metrics of accuracy, precision, recall, F1 score, and AUC-ROC from the use of the optimized model on the test dataset and the initial model on the training dataset to evaluate the performance of the optimized model.

The accuracy is the proportion of all correct predictions (positive and negative) out of the total predictions (DataCamp, 2024). The accuracy of my model after hyperparameter tuning increased from 86.2% to 89.0%. The precision measures the quality of all positive predictions by measuring all instances of positive predictions and determining the proportion of those that actually were positive (Google Developers, 2025). This metric decreased from 85.7% to 80.7% after hyperparameter tuning. The recall measures all actual positive instances and determines how many the model correctly identified (Google Developers, 2025). This metric increased significantly from 58.2% to 75.0%. The F1 score is the mean between precision and recall. This also had a notable increase from 69.3% to 77.7%. The AUC-ROC represents the probability that a randomly chosen positive incident will be ranked higher than a randomly chosen negative one (Scikit-learn developers, 2024). This metric also had a significant increase from 77.3% to 93.7%.

2. Discuss the results and implications of your classification analysis.

The recall after hyperparameter tuning increased significantly, meaning that the proportion of correctly identified churners increased. Even though the precision slightly decreased, it remained strong, indicating that positive predictions are still reliable. The overall accuracy and F1 score improved, showing a better balance between precision and recall. The substantial improvement of the AUC-ROC score confirms the model's ability to distinguish churners from non-churners.

3. Discuss one limitation of your data analysis.

One limitation of the data analysis is class imbalance, which means that churners are less frequent than non-churners. Accuracy can be misleading, so recall and F1 score metrics become more critical.

4. Recommend a course of action for the real-world organizational situation from part B1 based on your results and implications discussed in part F2.

After hyperparameter tuning, my model is substantially better at identifying actual churners. Now, telecommunications companies can more confidently reach out to customers who are predicted to churn with retention offers. I recommend regularly flagging high-risk customers with proactive retention campaigns and monitoring model performance.

REFERENCES

DataCamp. (2024). *Random Forest Classification in Python*. https://www.datacamp.com/tutorial/random-forests-classifier-python

DataCamp. (2024). *Hyperparameter Tuning in Python*. https://www.datacamp.com/tutorial/hyperparameter-tuning-python

Google Developers. (2025). *Classification: Accuracy, recall, precision, and related metrics*. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Scikit-learn developers. (2024). *RandomForestClassifier*. In scikit-learn 1.7.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Scikit-learn developers. (2024). *GridSearchCV*. In scikit-learn 1.7.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Scikit-learn developers. (2024). *Metrics and scoring: quantifying the quality of predictions*. In scikit-learn 1.7.2 documentation. https://scikit-learn.org/stable/modules/model_evaluation.html