**Flowchart Of Data Through My Pipeline**

EDA (Exploratory Data Analysis)

Convert Datetime Objects To Features (Novel)

Label Encode Useful Text Columns

Produce Final Results

Model Evaluation

Model Testing

Further EDA (Exploratory Data Analysis)

Drop Redundant Columns

Balance Classes By Undersampling

Encode FRAUD-NONFRAUD

Read in Data

General overview of my approach:

Pretty much all machine learning problems include the following steps: data cleaning, exploratory data analysis, model training, model evaluation. This project was no different.

My approach to EDA and data cleaning began by trying to understand the format of the data. What were the datatypes, was their spread like per column? I learned that there was a large degree of variation in some of the columns and that there was a variety of data types represented. This informed first that I would like to see later if there was correlation between different features which was plotted using a heatmap, and then that I would like to encode a lot of these features as numerical values if possible or worth-while. I noticed when evaluating the dataframe that there were some redundant datetime columns, so they were dropped. Additionally, I plotted the class distribution so that I could determine if there was imbalance. I learned that the classes were imbalanced which caused me to under sample the majority class (FRAUD). I did this because in many cases, imbalanced classes can negatively impact the performance of your model. I noticed that a lot of the datetime features would likely be useful in determining fraud was occurring, so I developed an approach to break datetime objects into component year, day, month, hour, min, second components. I had to account for incorrectly formatted datetime objects, as well as null values (which were replaced by impossible datetime values). This improved the performance of my models by nearly 10% when all was said and done.

Once EDA and data cleaning was complete it was time to train and evaluate models. I tested many models, including some that I did not include in my jupyter notebook. These models not included performed so poorly that they were removed. Models that I chose to evaluate and keep included RandomForest, SVC, MLPClassifier, XGBoost. These models all performed pretty well. The two best performing models were RandomForest and XGBoost. I attempted to perform an ensemble method known as Voting Classifier that uses multiple models to make classifications but I did not see any improvement in F1 from doing this so I dropped the idea. Models underwent hyperparameter tuning through GridSearchCV. This enabled me to determine the optimal hyperparameter values for this particular problem. Most of the Models that I tested were not neural networks and therefore I did not feel the need to use RandomSearchCV as the results would have likely been poorer using this approach. Models were evaluated through their confusion matrix, classification report, ROC AUC and AUC-PR. I additionally plotted the ROC curves and precision-recall curves for each model to illustrate their performance.

Citations (APA7)

Brownlee, J. (2021, January 12). *How to USE Roc curves and PRECISION-RECALL curves for classification in Python*. Machine Learning Mastery. Retrieved September 23, 2021, from https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.

*Dealing with Datetime features in Python and pandas*. Analytics Vidhya. (2020, May 5). Retrieved September 23, 2021, from https://www.analyticsvidhya.com/blog/2020/05/datetime-variables-python-pandas/.

Nelson, D. (2019, May 11). *Overview of classification methods in Python with scikit-learn*. Stack Abuse. Retrieved September 23, 2021, from https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/.

Panjeh. (2020, June 29). *Scikit learn hyperparameter optimization FOR MLPCLASSIFIER*. Medium. Retrieved September 23, 2021, from https://panjeh.medium.com/scikit-learn-hyperparameter-optimization-for-mlpclassifier-4d670413042b.

*sklearn.preprocessing.LabelEncoder¶*. scikit. (n.d.). Retrieved September 23, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.

Sondosomar. (2021, September 18). *Credit card fraud team*. Kaggle. Retrieved September 23, 2021, from https://www.kaggle.com/sondosomar/credit-card-fraud-team.

Steen, D. (2020, September 20). *Precision-Recall curves*. Medium. Retrieved September 23, 2021, from https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248.