Customer Churn Prediction

Udacity Machine Learning Engineer Nanodegree

Jin Kwon

Re-submitted Jan. 17th, 2020

1. Introduction

1.1.Project Overview

Customer base is a company's primary source of business and revenue. Similarly, the loyalty of a customer base determines the success of a company. Given that, it is not surprising that many companies have teams specialized in customer analytics that models and predicts many facets of customers including customer lifetime value and customer segmentation.

One important topic related to customer analytics is understanding and predicting customer churn. According to HubSpot[1], customer churn rate is defined as "the percentage of your customers or subscribers who cancel or don't renew their subscriptions during a given time period." A certain "healthy" rate of customer churn is expected and unavoidable. For example, a company cannot take actions to keep customers who are leaving a service due to no longer needing it (e.g., sold a car and not needing car insurance) or moving to another country.

For many services, however, reasons for customer churns are often something that can be avoided if a company intervenes and takes an action. Moreover, often investing in keeping a customer (avoiding churn) is significantly lower compared to revenue lost due to churn (revenue churn) and consequently acquiring new customers to fill in the churned customers.

In this project, we applied machine learning to build a model that predicts and gives insights into understanding customer churn, using Telco's customer churn data[2].

1.2. Problem Statement

The main problem that was tackled in this project is twofold. First, a machine learning model that can be deployed in the hypothetical business case was be developed, which predicted whether a customer will churn given a set of features which include payments, demographics, and subscribed service types. Models were selected such that it covers both the rather more traditional statistical models (e.g., logistic regression) as well as more recent ensemble-based models (e.g., XGBoost). More details on the models are discussed in *Section 2.1.1*. Second, the feature weights of the best performing model was analyzed for insights into features that are highly related to customer churn.

The result of the project are the following 1) a machine learning model and its best performing data augmentation method, and a set of hyperparameters that accurately predict the binary label (churn or no-churn) and 2) analysis results on the feature weights from the best-performing model, which can help the Telco company to make the best customer support decisions.

1.3. Metrics

Metrics were carefully selected to cover two main points. First was dealing with the skewed customer churn label. For most normally operating companies, customer churn is a rare event. In the Telco customer churn data, the number of churns (positive) label was fewer than the no-churn (negative) label. Specifically, there were approximately ~2k positive (churned) labels and ~5k negative (non-churned) labels. Given that, well-known metrics such as precision (number of correct/number of all labels) will be misleading because we can already achieve high precision by predicting all labels as the majority label; with our data, we will already achieve a precision of ~60% predicting everything as no-churn. Thus, AUC-ROC was used as an accuracy metric that captures how much the model can distinguish between the positive and negative classes.

Second, the assumption that we made in the project is that for a company, revenue loss due to customer churn is significantly larger (revenue loss + investment for new customers) than maintaining a customer from churning. Given that, looking at false positives and false negatives was deemed to be valuable for companies to make revenue-related decisions. Thus, TP rates and FP rates were summarized in this report as well.

2. Analysis

2.1. Data Exploration & Exploratory Visualization

4 Numerical, 16 Categorical, and 1 label columns were available. However, since the `SeniorCitizen` numerical column was being used as a binary label, this was updated to a categorical binary label “Yes” and “No” to be consistent with other binary labeled categorical features.

Three numerical columns were `tenure` in months, `MonthlyCharges` in money unit per month, and `TotalCharges` in money unit. One immediate finding from this was that `TotalCharges` can be expressed in terms of `tenure` and `MonthlyCharges`. As can be seen from **Figure 1**, a scatter plot of `TotalCharges` and `MonthlyCharges*Tenure` resulted in a nearly straight line. Pearson correlation coefficient also resulted in ~0.9996, which indicated that the result is nearly perfectly correlated. Since `TotalCharges`, `MonthlyCharges`, and `Tenure` are linear combinations of each other, `TotalCharges` column was dropped because the information from this column could be expressed in terms of the remaining two. This also took care of the issue with the only column having a N/A, the 11 N/As in the `TotalCharges` column.

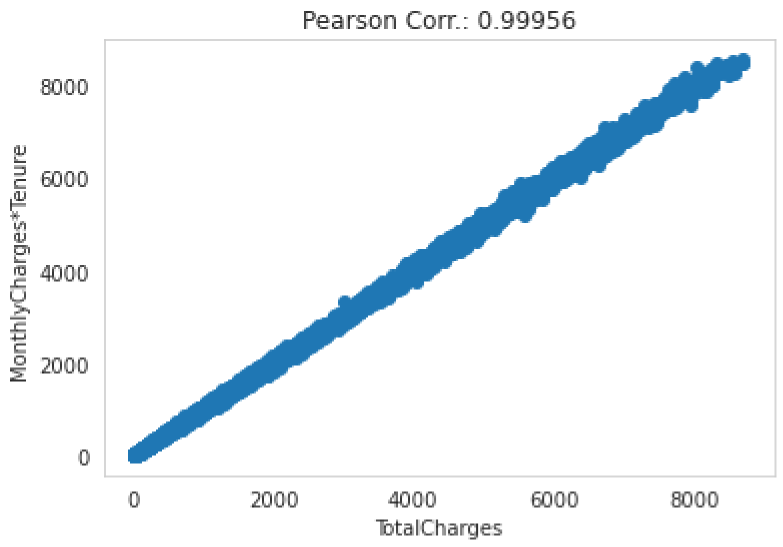


Figure 1 – Scatter plot between `TotalCharges` and `MonthlyCharges*Tenure`

Categorical variables in Telco could be categorized into three main groups based on their characteristics: service-related (mainly phone and internet), demographics-related, and payment-related. Similar to how we could easily remove the highly correlated numerical column, high correlation in categorical columns was observed from the contingency table. For example, a non-no value of `MultipleLines` already indicated a `Yes` to PhoneService as can be seen from **Table 1**.

Table 1 – Mutually Exclusive Categorical Columns

	PhoneService:No	**PhoneService:Yes**
InternetService:No	0	3390
InternetService:No phone service	682	0
InternetService:Yes	0	2971

Other sets of categorical features indicated some correlations looking at the contingency table. However, using statical tests such as Fisher's Exact Test showed that these columns are not statistically significantly using Fisher's Exact Test. Please refer to Appendix **Section A** for details on the numerical and categorical variables.

2.2. Algorithms and Techniques

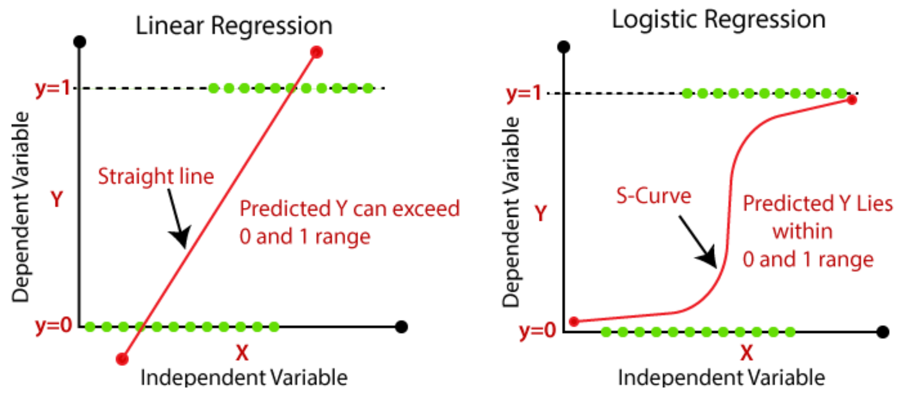
In this section, a brief explanation of the different candidate machine learning models and under/over sampling techniques are explained.

2.2.1. Machine Learning Algorithms

Models used in this project are all proven to perform well for binary classification tasks. However, there are minute differences across models such as how each model makes predictions. The models can be categorized into two main groups: more "traditional" statistical models and more recent ensemble-based models.

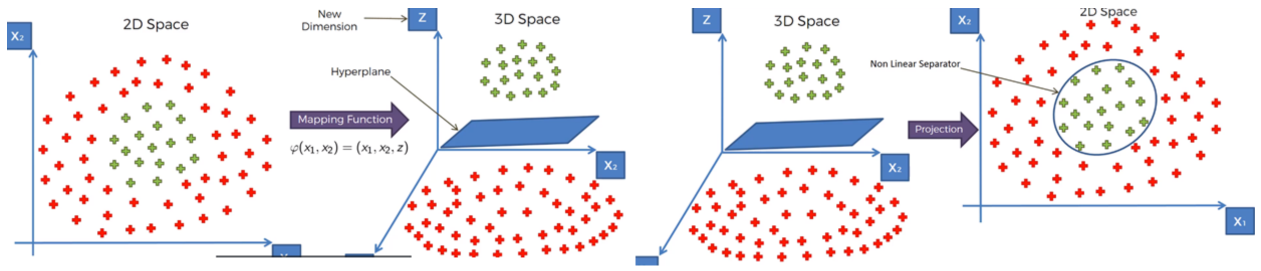
First, two models in the "traditional" model bucket are logistic regression and support vector machine (SVM). Logistic regression is a simple regression-based model that utilizes logistic function to output binary predictions given numerical features (**Figure 2**). The biggest appeals of using logistic regression are 1) accommodates both numerical and categorical features, 2) computationally inexpensive to train and 3) results are highly interpretable (thus, more "business-friendly") compared to other ensemble-based models. However, a major limitation with the logistic regression is that it assumes linearity between the dependent and the independent variables. Thus, if there is any non-linear relationship between the features of Telco data to the churn label, logistic regression will have a difficulty capturing the relationship.

Figure 2[3] – Linear Regression vs. Logistic Regression



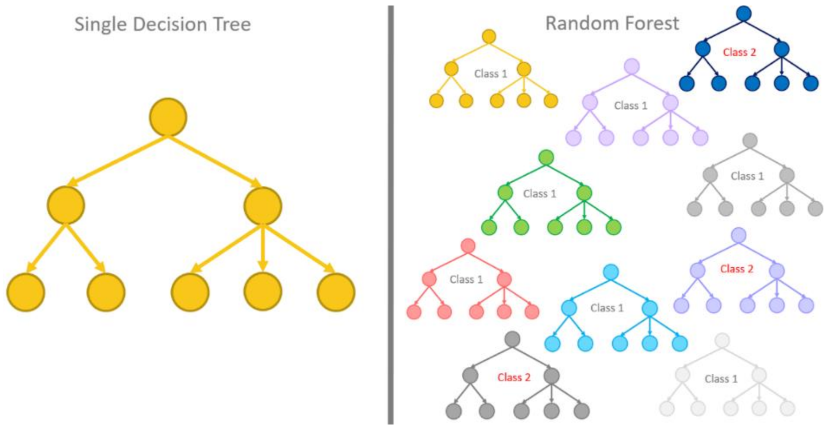
SVM utilizes a subset of data points that influence the separation of class labels (support vectors). The default hyperplane is linear, similar to that in logistic regression. However, using kernel tricks, which is a process of mapping features into a higher dimension as in **Figure 3**, linear decision boundary can deal with non-linearity. As one can imagine, finding the optimal kernel function to increase prediction accuracy with SVM can be a challenging task.

Figure 3[4] – Kernel trick in SVM



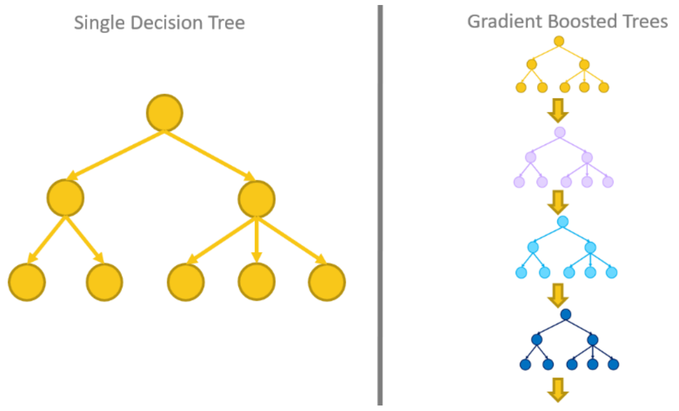
The second set of models are ensemble-based machine learning models which often resulted in high prediction scores in Kaggle competitions[5]. The ensemble models used in this project can again be categorized into two: Random Forest and Boosting models. Random Forest model, simply put, can be thought of as collecting results from many decision trees (i.e., forest) and applying a clever aggregation method to utilize the "wisdom of the crowd" of decision trees (**Figure 4**).

Figure 4[6] – Random Forest Model Diagram



Whereas the random forest model builds many decision trees and aggregate the result, boosting methods sequentially build models in such a way that the next model learns predictive patterns from the remaining noise (variance) from the previous model. You can think of this as "trying to squeeze out more juice at a step".

Figure 5[6] – XGBoost Model Diagram

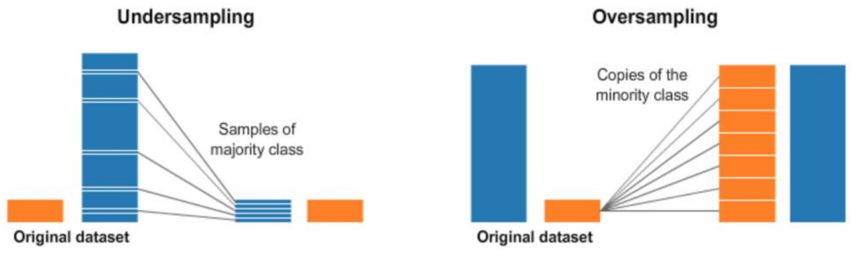


CatBoost and XGBoost are similar at its core, with CatBoost having additional benefits in dealing with categorical features and optimizations in the training process[7]. The high performance of ensemble models comes at the cost of being computationally expensive.

2.2.2. Under-sampling and Over-sampling

As mentioned earlier, class imbalance was one of the main challenges in the current project. There are two well-known practices to tackle the class imbalance problem. The first one, under-sampling, refers to decreasing the size of the majority class to the size similar to the minority class. The benefit here is that since we have nearly the same number of data points for the positive and negative classes, the model will not bias towards predicting the majority class. However, one major drawback is that we are throwing away valuable data for the sake of balancing out the data size.

Figure 6 – Under Sampling and Over Sampling



What we can do to keep the sizes of the two labels similar, while not throw away valuable labelled data is to increase the number of the minority class. Although making same copies of the minority class data could work, there is a cleverer statistical methods which is the Synthetic Minority Over Sampling Technique (SMOTE). SMOTE generates synthetic data points by generating the most probable feature values, given the existing real feature values in the data. SMOTE-NC can be thought of as an extension of SMOTE, with some additional features to augment categorical features.

In this project, four up-sampling and down-sampling methods were adopted. For down-sampling, the majority class was randomly down-sampled so that the number of majority class either 1) matches the number of a minority class or 2) be 1.5 the size of the minority class. For up-sampling, the minority class SMOTE and SMOTE-NC methods were adopted. Unlike the SMOTE method, SMOTE-NC allows up-sampling data with categorical features.

2.3.Benchmark

A logistic regression model with one-hot encoded training data set was used as a benchmark model. However, hyperparameter tuning was conducted to get the best AUC-ROC score for the logistic regression. The best hyperparameters were solver: newton-cg, penalty: L2, C: 0.1, and the resulting AUC-ROC was 0.823.

3. Methodology and Implementation

In this section, we will go through the data preprocessing and modeling pipeline.

3.1.Data Preprocessing

Multiple steps of data preprocessing were conducted. The first was removing highly correlated features. As discussed in *Section 2.1.*, *`TotalCharges`* was removed for being highly correlated with the multiplication of `Tenure` and `MonthlyCharges`. This naturally took care of the issue with NA. Secondly, `PhoneService` column was removed as it was completely explainable by the `InternetService` column.

Secondly, outliers were checked using histograms for the numerical values and concluded that there were no obvious ones (**Figure 7** and **Figure 8**). Additionally, the values of tenure and monthly charges were within acceptable ranges; for example, there was no nonsensical monthly charge value such as 360,000. Thus, there was no need to apply more advanced methods, such as Cook's Distance.

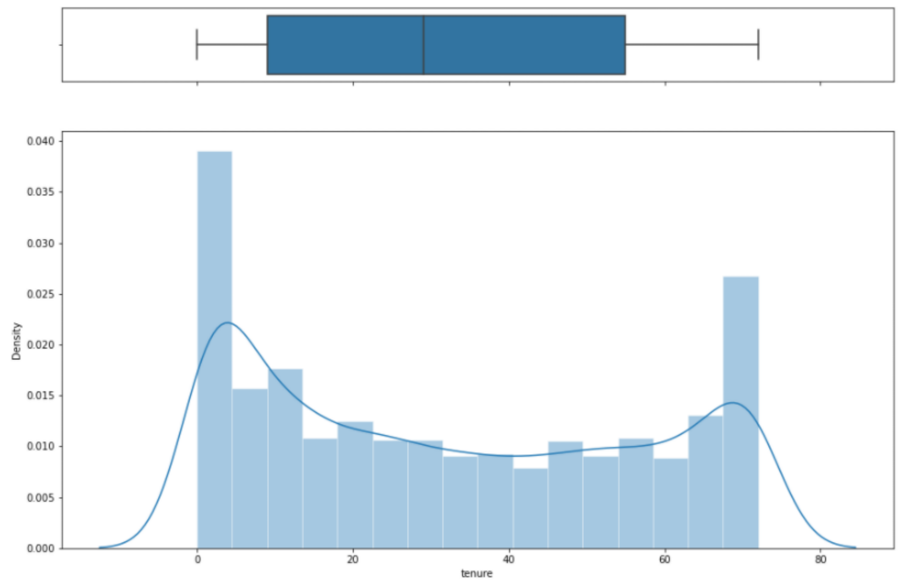


Figure 7 – Distribution of `tenure`

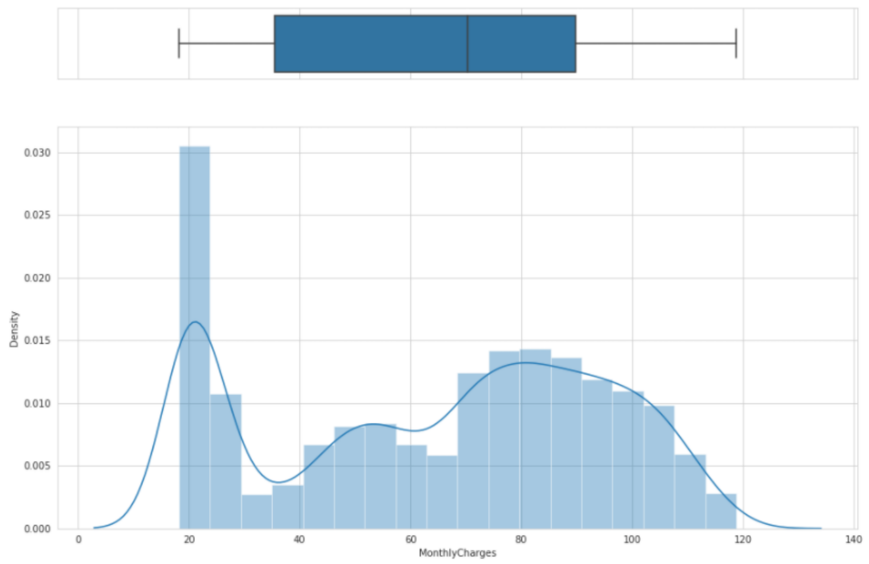


Figure 8 – Distribution of `MonthlyCharges`

3.2. Train/Test Split and Data Augmentation

Once the data have been cleaned up, the data was split into training and testing sets using the train_test_split() method from the scikit-learn package. During this process, the splitting process was stratified by the label `churn` class because we wanted to have similar fractions of positive and negative labels in both training and testing sets. Following that, categorical features were one-hot encoded to accommodate models that only take numerical features.

Then, four different data augmentation methods were applied to the testing data set. First, the majority class training data was under-sampled (sampling_strategy=’majority’) to have an identical size as the minority class. Second, the majority of class training data was under-sampled to be 1.5 the size of the minority class (sampling_strategy=0.667). Third, scikit-learn’s SMOTE with default hyperparameters was used to up-sample the minority class to the same size as the majority class. Lastly, SMOTE-NC was used to up-sample the minority class to the same size as the majority class.

3.3. Model Training

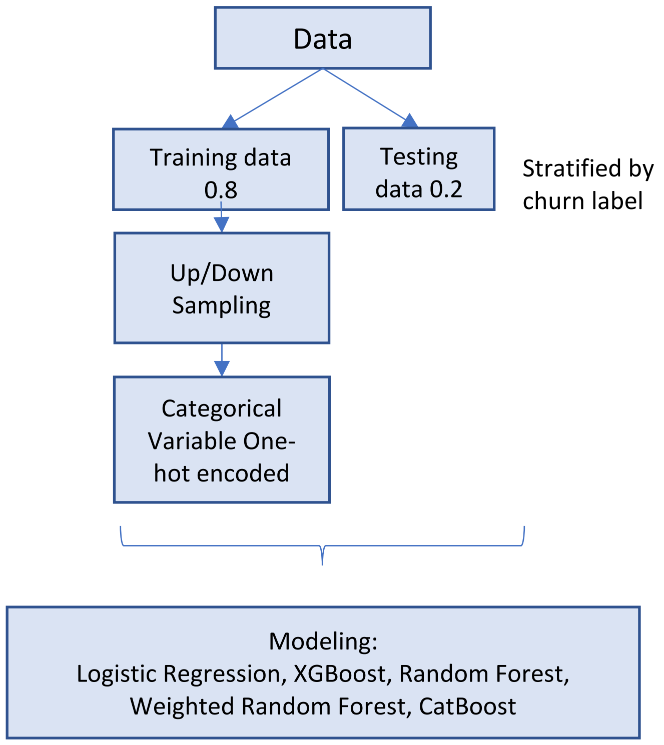
After the data was split into train/test sets, categorical values one-hot encoded, and data points up/down sampled, scikit-learn GridSearch() method was implemented with six different ML models. **Table 2** specifies the hyperparameter feature space used in the current project.

Table 2 – *Hyperparameter Search Space*

Model	**Hyperparameter Search Space**
Logistic Regression (benchmark)	• solvers = ['newton-cg', 'lbfgs', 'liblinear'] • penalty = ['l2'] • c_values = [100, 10, 1.0, 0.1, 0.01]
Random Forest and Weighted Random Forest	• n_estimators = [10, 100, 1000, 1200, 1500] • max_features = ['auto', 'sqrt'] • max_depth = [None, 2, 4, 10, 12, 20, 30] • min_samples_split = [2, 5, 10] • min_samples_leaf = [1, 2, 4] • class_weight = [{1: .9, 0: .1}, {1: .8, 0: .2}]*
XGBoost	• n_estimators = [10, 100, 1000, 1200] • learning_rate = [0.001, 0.01, 0.1] • subsample = [0.3, 0.5, 0.7, 1.0] • max_depth = [2, 3, 7, 9]
Support Vector Machine	• kernel = ['poly', 'rbf', 'sigmoid'] • C = [80, 50, 10, 1.0, 0.1, 0.01]
CatBoost (one-hot encoded)	• iterations = [2, 4, 10, 12, 24] • learning_rate = [0.001, 0.01, 0.1, 1] • depth = [2, 3, 7, 9]

* only applicable to weighted random forest

The resulting data preprocessing, augmentation, and modeling pipeline is presented in **Figure 9.**



Data
Training data 0.8
Testing data 0.2
Stratified by churn label
Up/Down Sampling
Categorical Variable One-hot encoded
Modeling:
Logistic Regression, XGBoost, Random Forest, Weighted Random Forest, CatBoost

Figure 9 – Data Preprocessing and Modeling Pipeline

The best AUC-ROC, TP rate, and TN rates, along with the best set of hyperparameters and up/down sampling methods were recorded for comparison across models. Results are reported in *Section 4.1.1*

3.4.Challenges and Solutions

One challenge with running the above pipeline was expensive computations. As a solution, which was already integrated with the above explanation, was to make a sparser hyperparameter search space. For example, a first run through the above pipeline indicated that the Random Forest model performed better with higher values for max_depth. In the next iteration, we decreased the size of search space by re-running the pipeline with fewer higher-range values for the max_depth hyperparameter. Secondly, switching the hardware to have more computing power from a laptop to a desktop with a better CPU significantly improved the computation speed.

4. Results

4.1.Model Evaluation and Validation

4.1.1 Model Predictability

Table 3 is the best AUC-ROC scores from each model, the best performing data augmentation methods, and the best set of hyperparameters. Both CatBoost models, one using one-hot encoded categorical features and the other using the raw categorical features, outperformed the rest of the models, even without the use of over/under sampling methods. Please note that data augmentation was not used for the CatBoost models due to time constraints.

Table 3 – *Model Comparisons*

Models	Best Resampling	Best Hyperparameters	AUC-ROC
Logistic Regression (base)	SMOTE-ENC	`{'solver': 'newton-cg', 'penalty': 'l2', 'C': 0.1}`	**0.823**
Random Forest	SMOTE	`{'n_estimators': 1500, 'max_features': 'sqrt', 'max_depth': 20, 'min_samples_split': 2, 'min_samples_leaf': 1}`	0.822
XGBoost	SMOTE	`{'n_estimators': 1000, 'learning_rate': 0.01, 'subsample': 0.7, 'max_depth': 9}`	0.828
Support Vector Machine	SMOTE	`{'kernel': 'rbf', 'C': 10.0, 'gamma': 'scale'}`	0.802
Weighted Random Forest	SMOTE	`{'n_estimators': 1500, 'max_features': 'sqrt', 'max_depth': None, 'min_samples_split': 2, 'min_samples_leaf': 1, 'class_weight': {1: 0.8, 0: 0.2}}`	0.817
CatBoost (one-hot encoded)	None*	`{'iterations': 24, 'learning_rate': 0.1, 'depth': 3}`	**0.846**
CatBoost (raw categorical variables)	None*	`{'iterations': 48, 'learning_rate': 0.1, 'depth': 3}`	0.845

** Due to time constraints, data augmentation was not applied with CatBoost model.*

Looking at the False Positive and False Negative rates of the best performing model (**Figure 10**), the result shows that the best performing CatBoost model did a great job predicting the majority negative class with a true positive rate of 0.85. However, the true negative rate was significantly lower with a score of 0.60. As future work, data augmentation methods used in the other models such as SMOTE will be used for CatBoost training, which can potentially improve the model's prediction on the minority class.

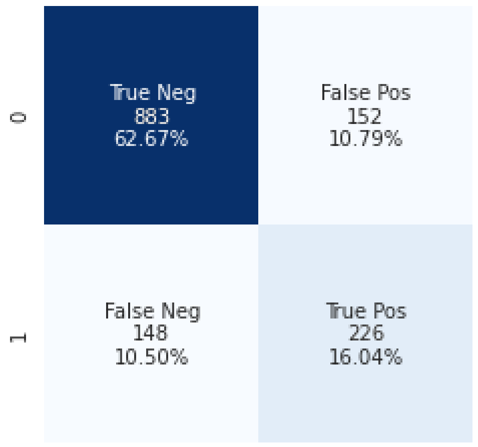


Figure 10 – Confusion Matrix

4.1.2. Feature Importance

The feature importance of `Tenure`, `MonthlyCharges`, and `Contract` far exceeded the rest of the features as can be seen in **Figure 11**. Given that `Contract`'s values are "Month-to-month", "One year", and "Two years", a rough conclusion can be drawn that churn is mainly dependent on time-related and money-related features, rather than the type of services the customers are signed up for or their demographic information. The conclusions are similar from XGBoost (**Figure A.3.**), Random Forest (**Figure A.2.**), and Weighted Random Forest (**Figure A.4.**). Future work could include understanding one-hot encoded CatBoost, which returned starkly different results as shown in (**Figure A.5.**).

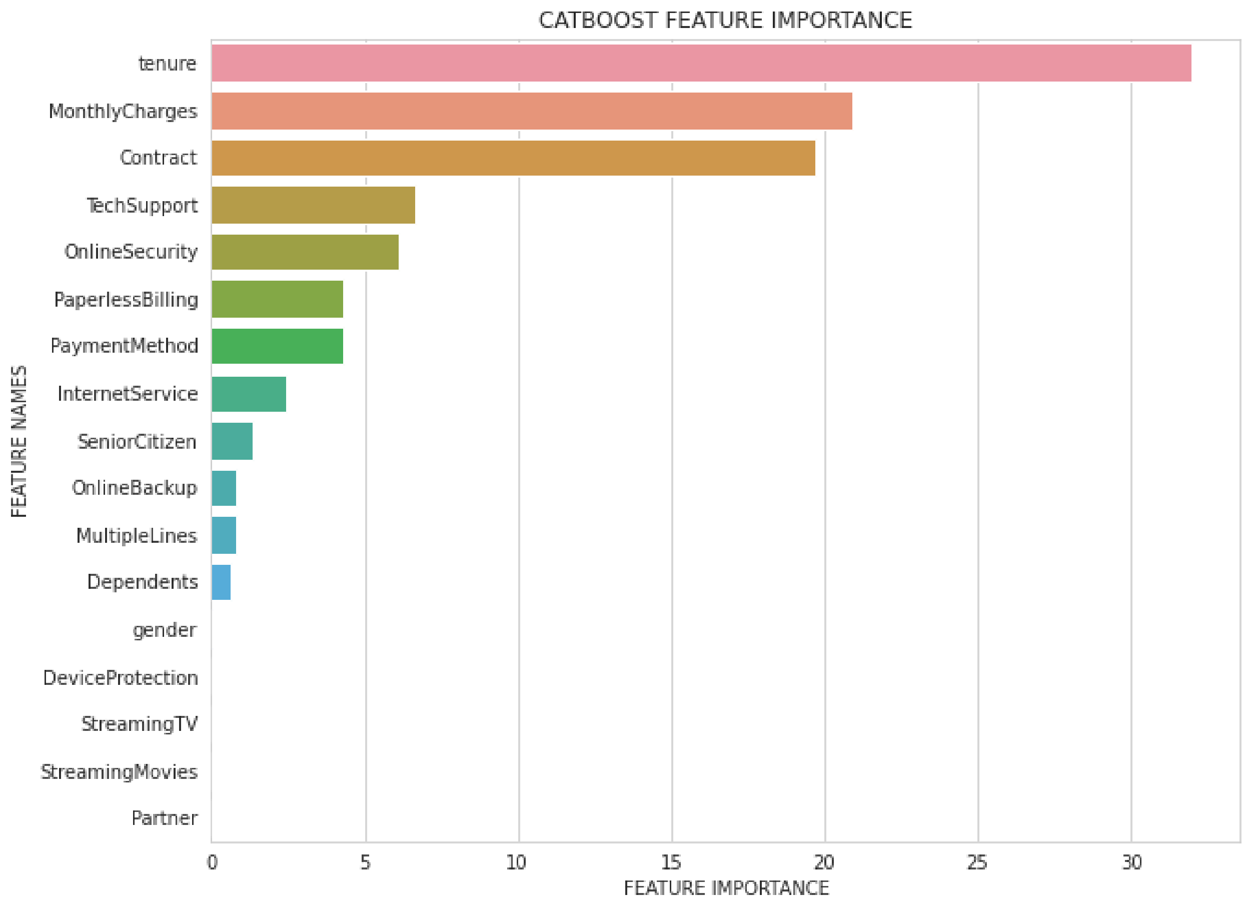


Figure 11 – Feature Importance of Best Performing CatBoost Model

4.2. Justification

The baseline model (logistic regression) and the best model (CatBoost) can be compared in two ways. The first is from using the AUC-ROC. The difference is 0.023 (0.823 vs. 0.846). Although the difference may not be significant, given that the hyperparameter selection has also been applied to the baseline logistic regression model, we can claim that the difference is significant. The second is comparing TP and TN rates. The TN rates were 0.73 for the baseline model and 0.85 for the best model. The TN rates were 0.74 for the baseline model and 0.6 for the best model. This result tells that the large gain in AUC-ROC of the best model resulted from correctly predicting the majority label and the minority (churn) class was more accurately predicted using the baseline model. Given that the baseline model used SMOTE oversampled data for training, we can be optimistic that similar performance gain in predicting the minority class can result in the best model once SMOTE is applied.

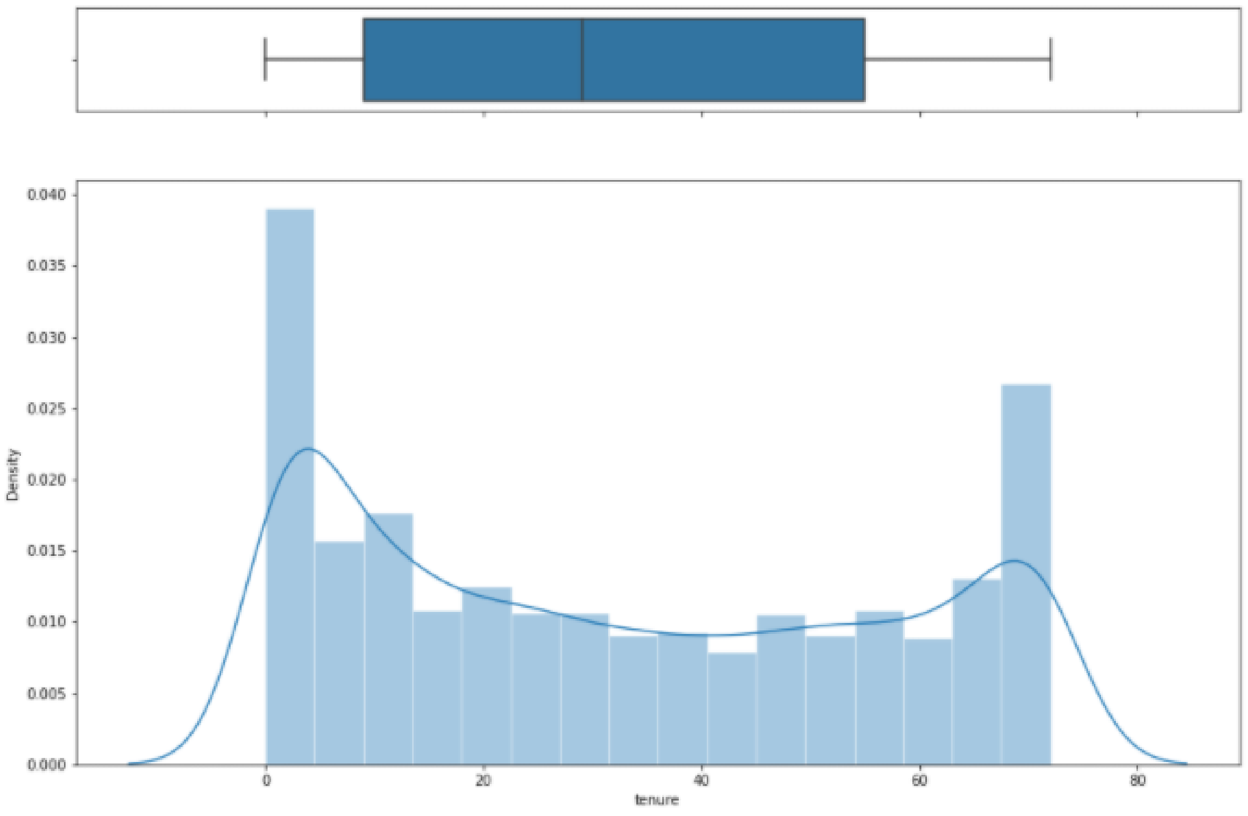
Reference:

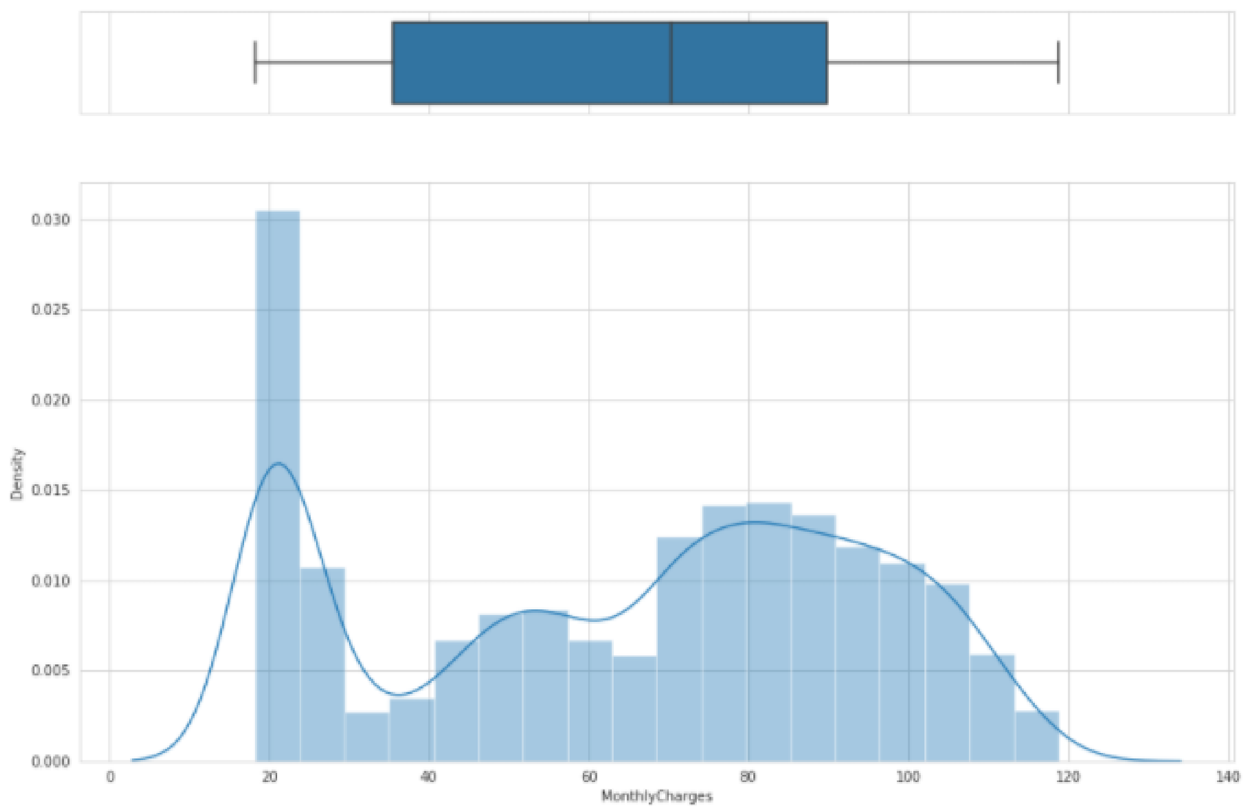
[1] Customer Churn Definition: What Is Customer Churn? [Definition] (hubspot.com)
[2] Telco Customer Churn data: Telco Customer Churn | Kaggle
[3] Linear and Logistic Regression: https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning
[4] The kernel in SVM: https://medium.com/pursuitnotes/day-12-kernel-svm-non-linear-svm-5fdefe77836c
[5] Kaggle Winning ML Solutions: Leaf Classification Competition: 1st Place Winner's Interview, Ivan Sosnovik | by Kaggle Team | Kaggle Blog | Medium
[6] Ensemble Model: https://medium.com/analytics-vidhya/ensemble-models-bagging-boosting-c33706db0b0b
[7] CatBoost Explanation: Categorical features - CatBoost. Documentation

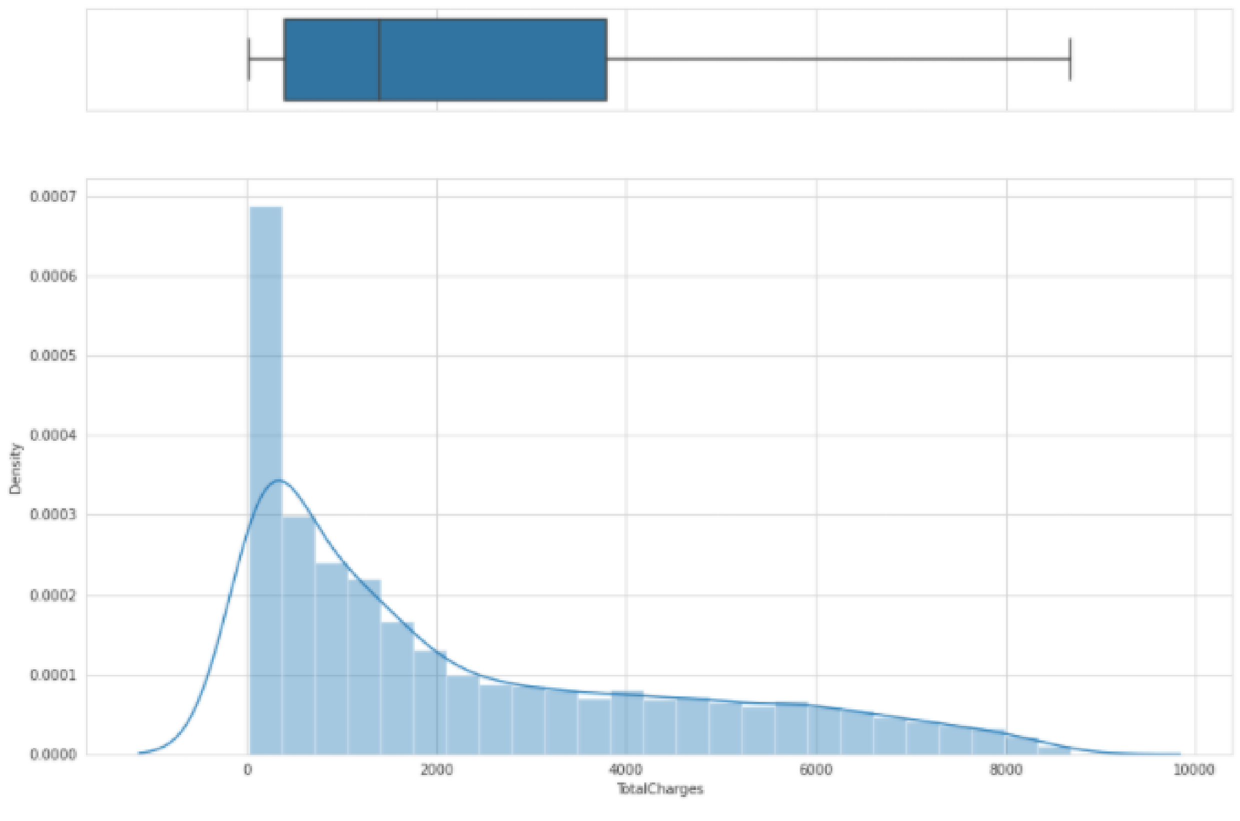
Appendix

A. Numerical and Categorical Features Distribution

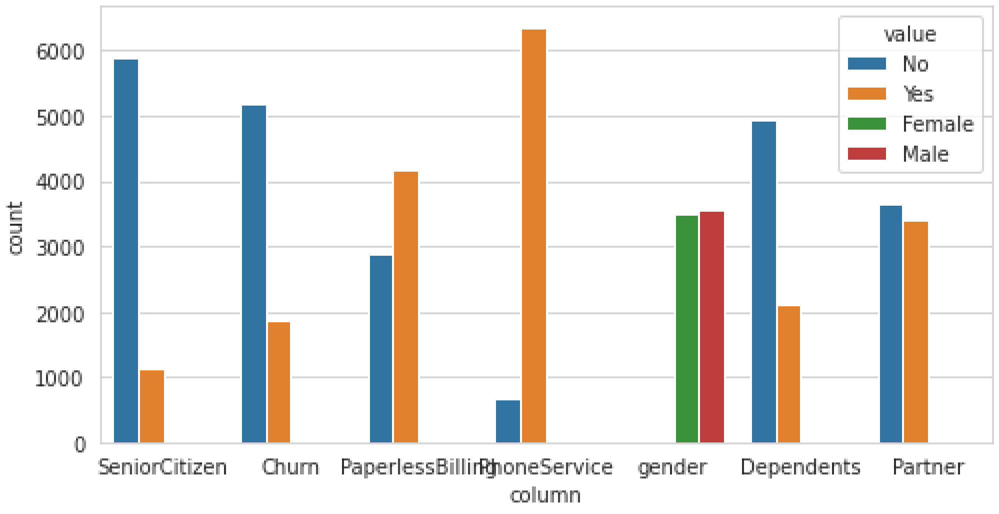
A.1. Numerical Features

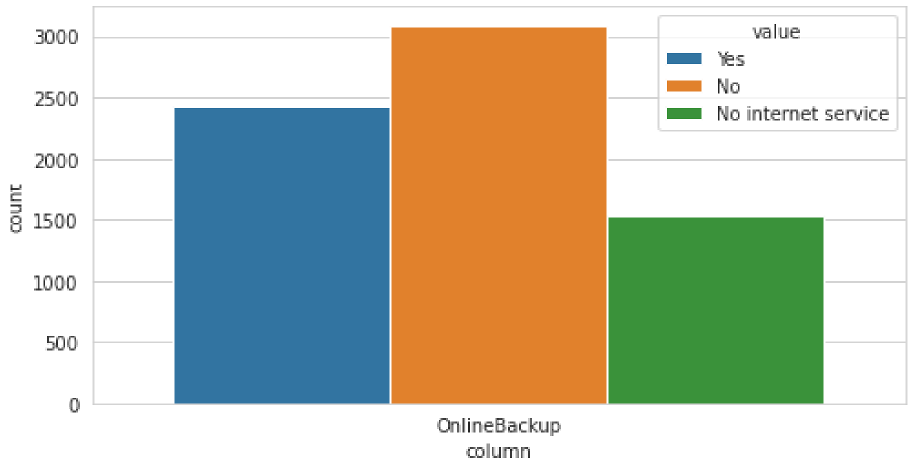


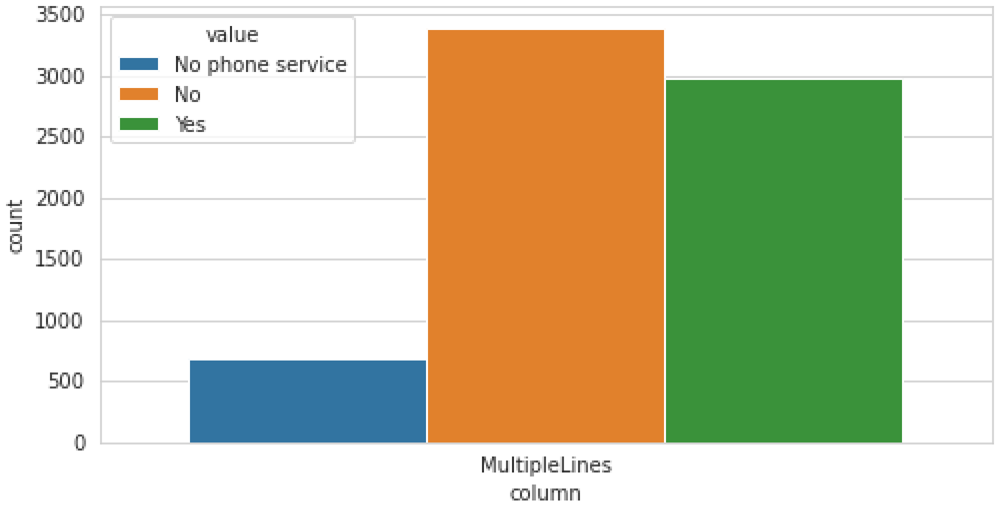

Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
120
140
MonthlyCharges

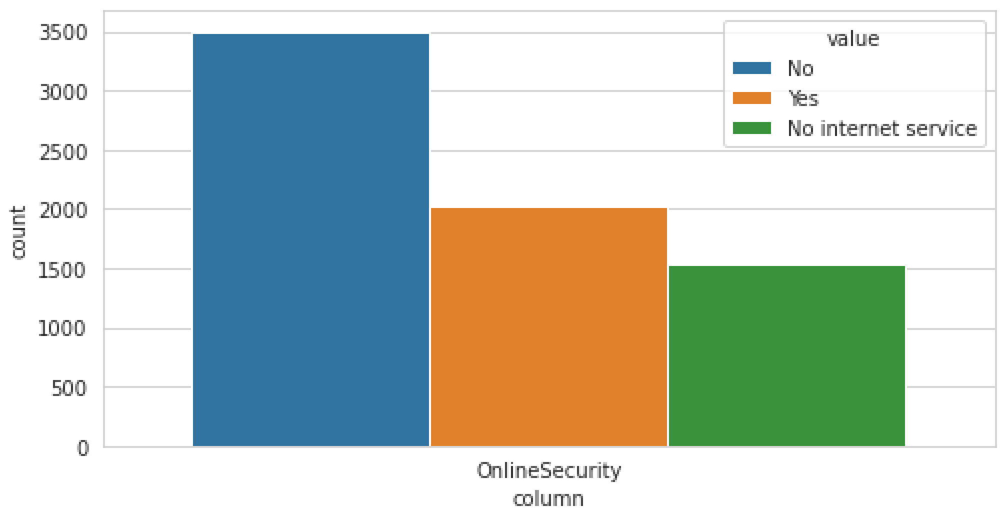

Density
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0
2000
4000
6000
8000
10000
TotalCharges

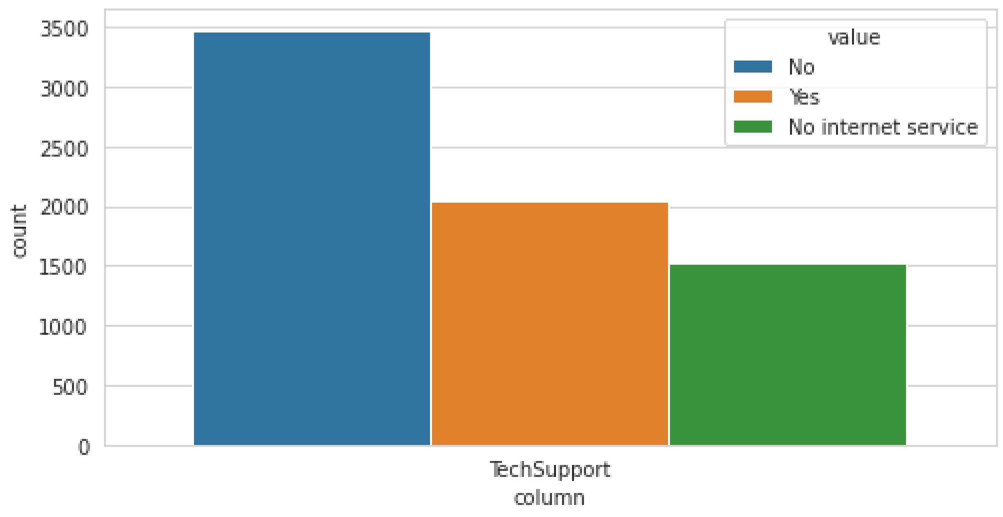
A.2. Categorical Features

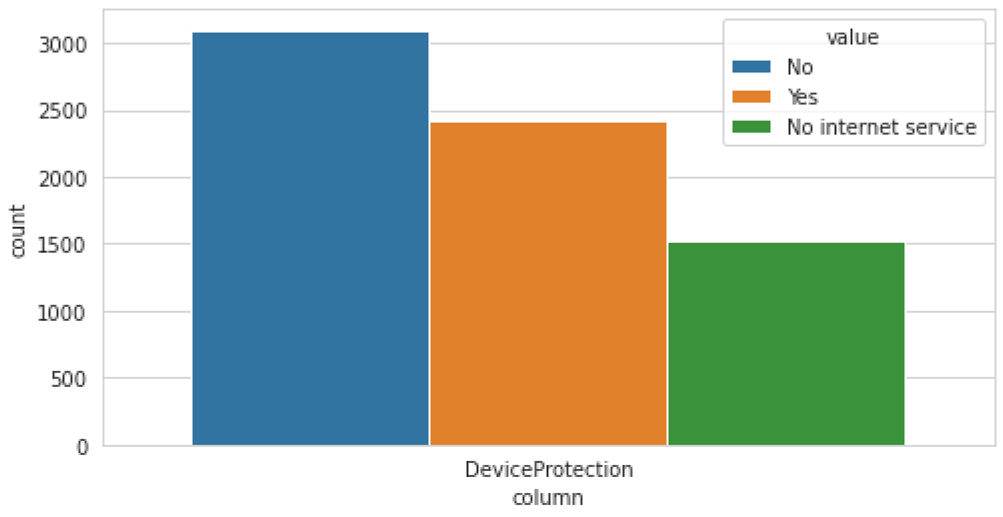


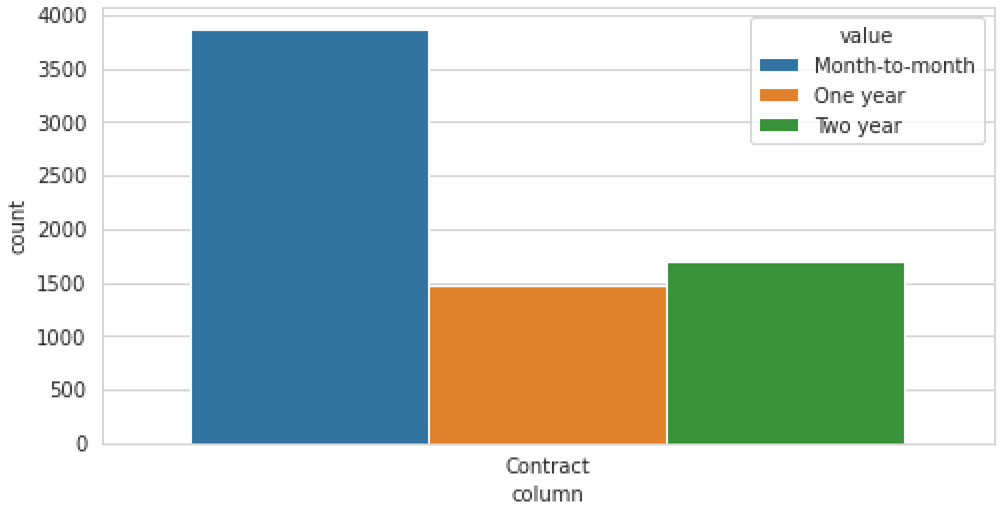




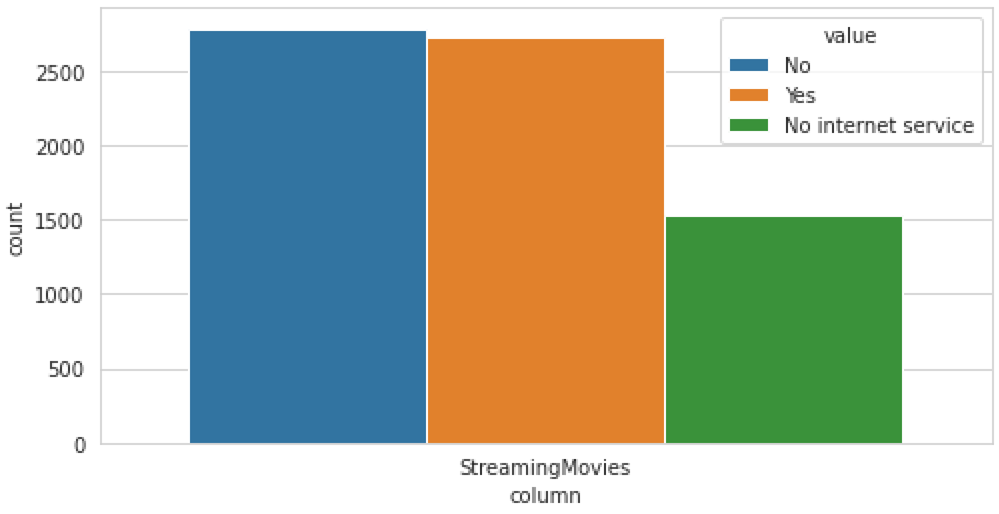

value
No
Yes
No internet service
count
0
500
1000
1500
2000
2500
3000
3500
OnlineSecurity
column


value
No
Yes
No internet service
count
0
500
1000
1500
2000
2500
3000
3500
TechSupport
column

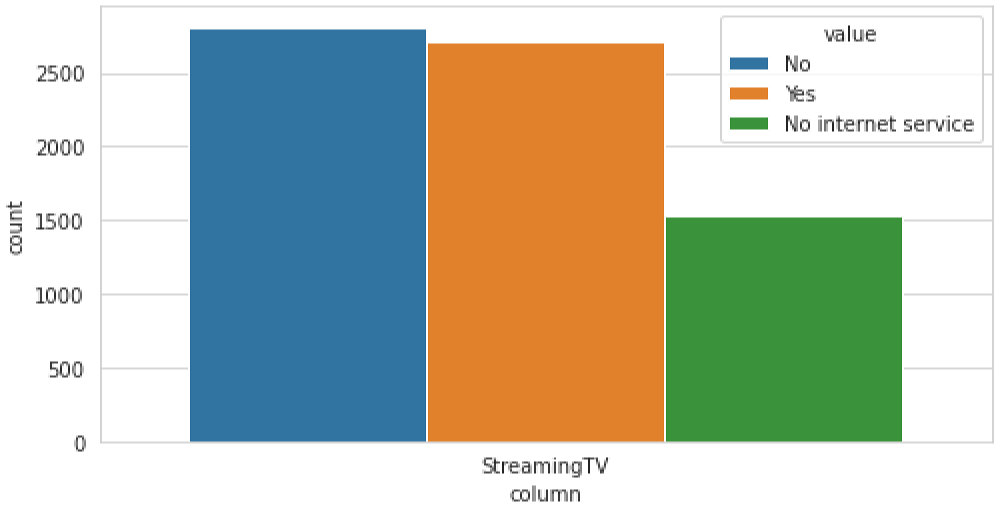

value
No
Yes
No internet service
count
0
500
1000
1500
2000
2500
3000
DeviceProtection
column



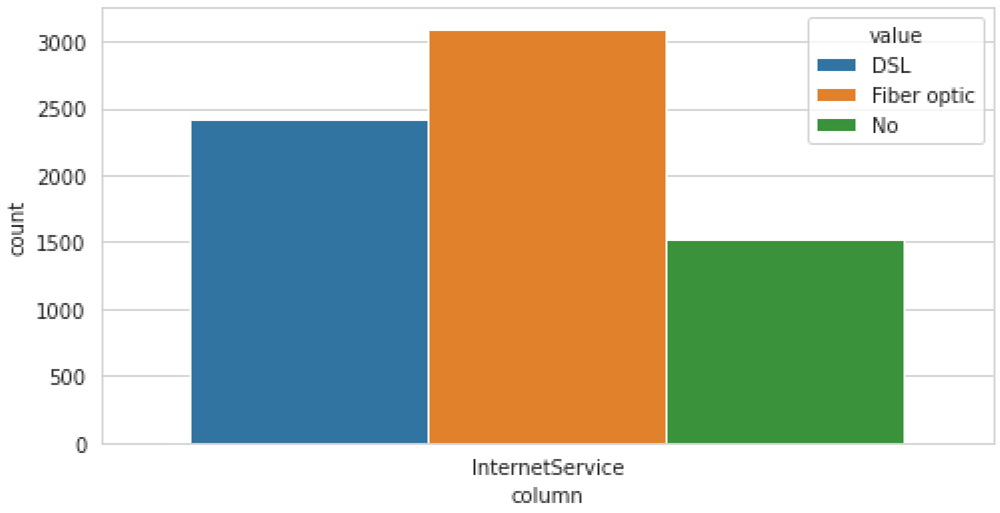
value
Month-to-month
One year
Two year
count
4000
3500
3000
2500
2000
1500
1000
500
0
Contract
column



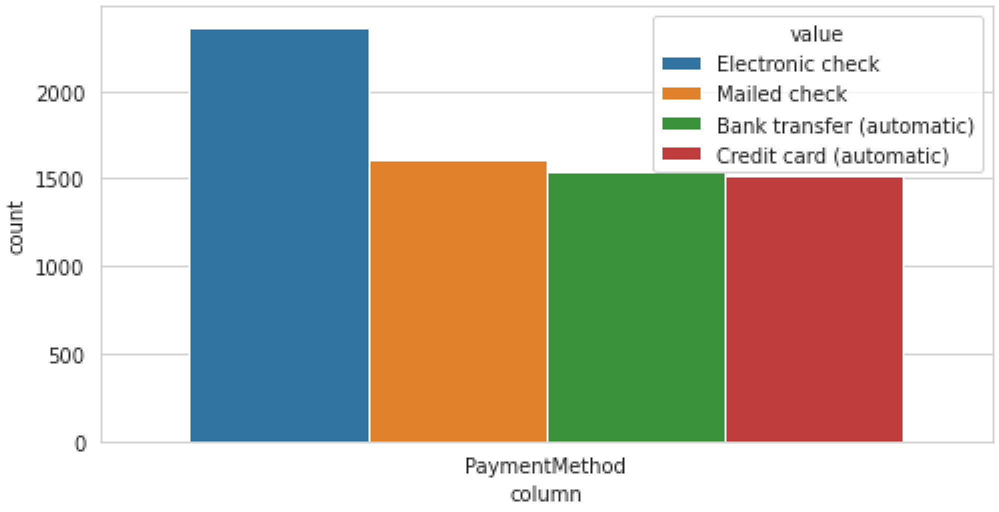
value
No
Yes
No internet service
count
2500
2000
1500
1000
500
0
StreamingMovies
column



value
No
Yes
No internet service
count
2500
2000
1500
1000
500
0
StreamingTV
column



value
DSL
Fiber optic
No
count
3000
2500
2000
1500
1000
500
0
InternetService
column



value
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
count
2000
1500
1000
500
0
PaymentMethod
column

B. Feature Importance

Figure B.1. – Logistic Regression Feature Weights

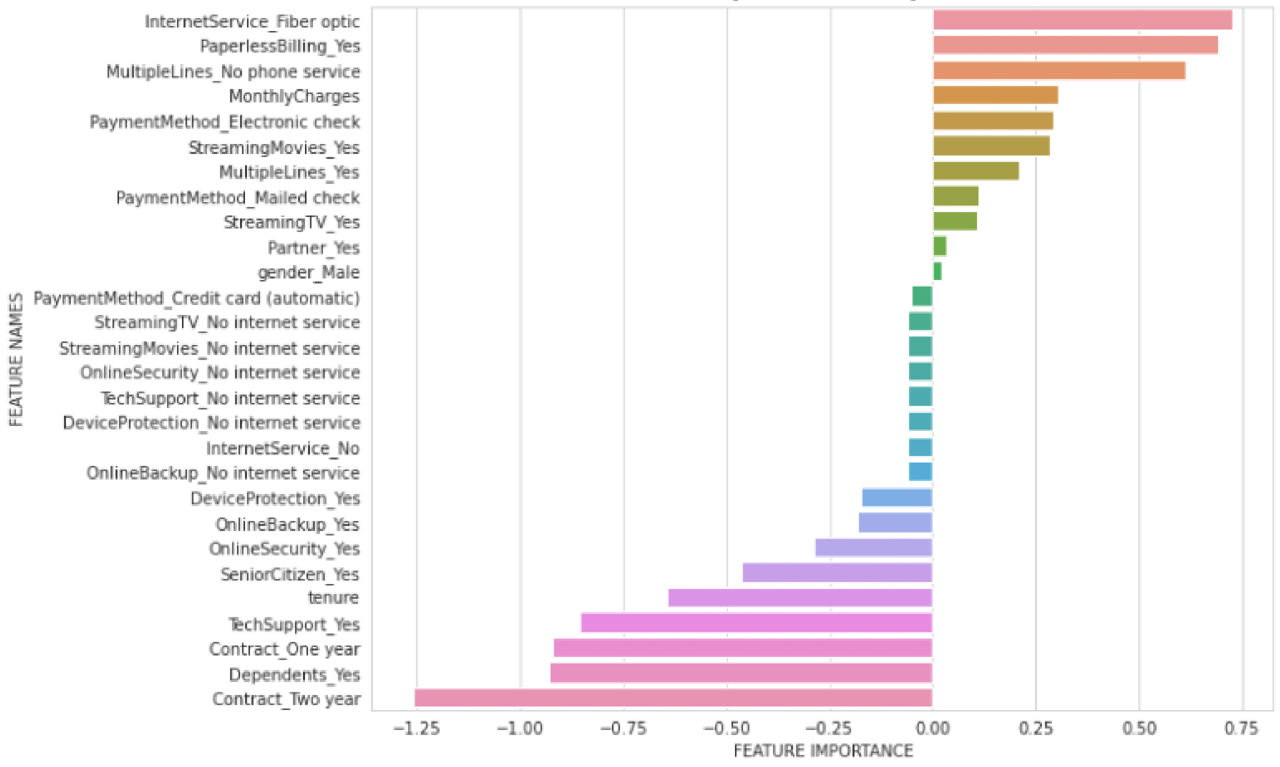


Figure B.2. – Random Forest Feature Weights

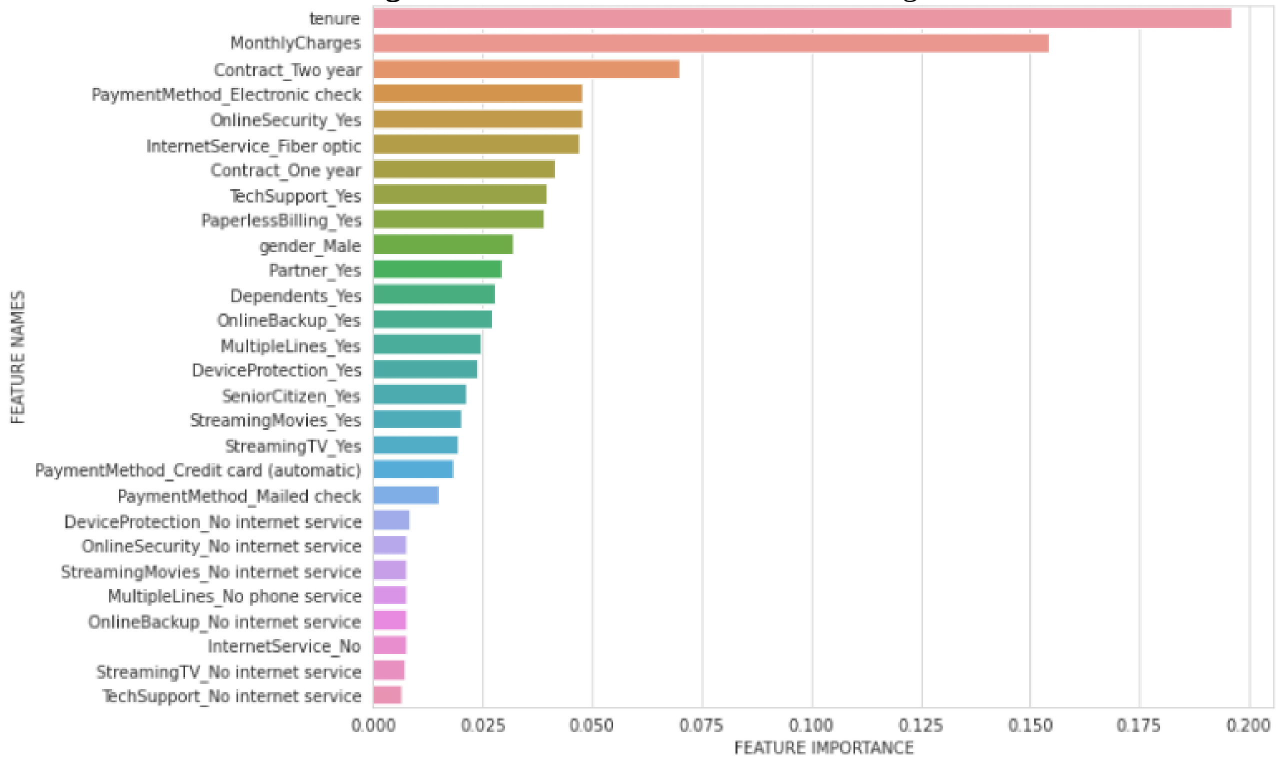


Figure B.3. – XGBoost Feature Weights

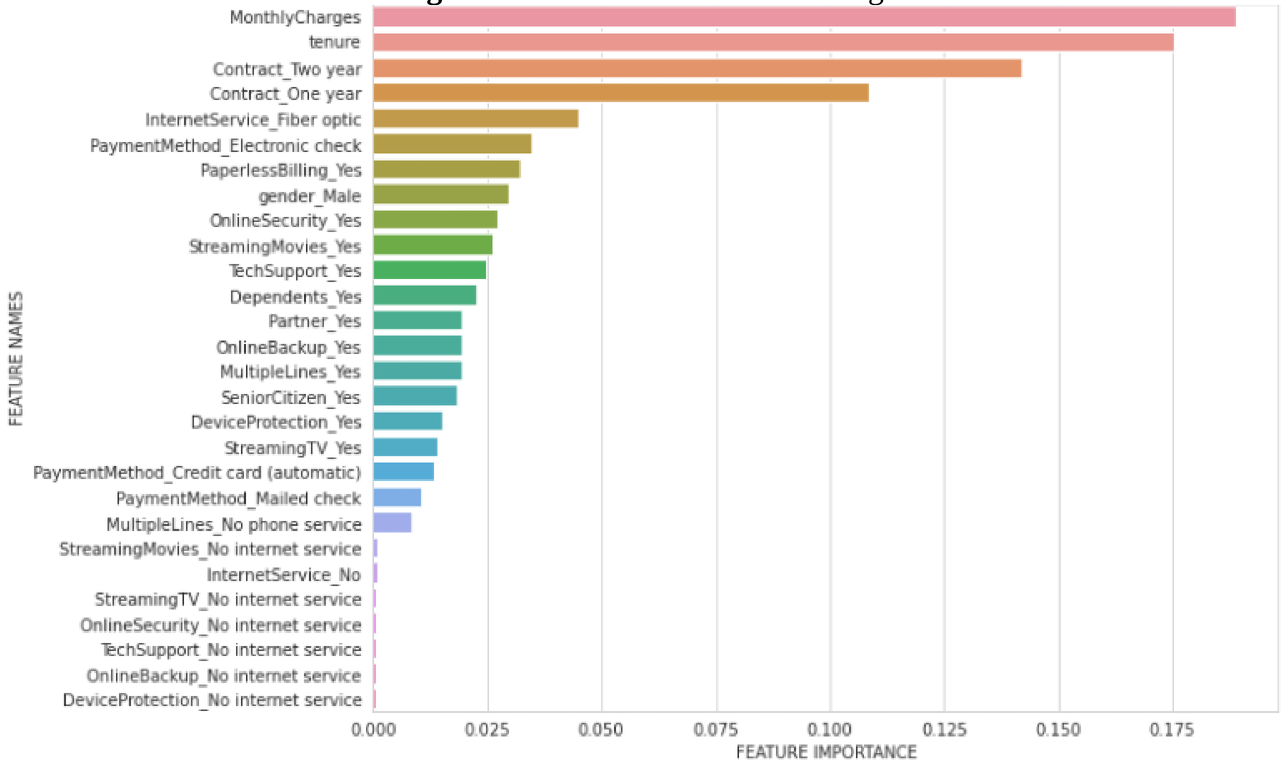


Figure B.4. – Weighted Random Forest Feature Weights

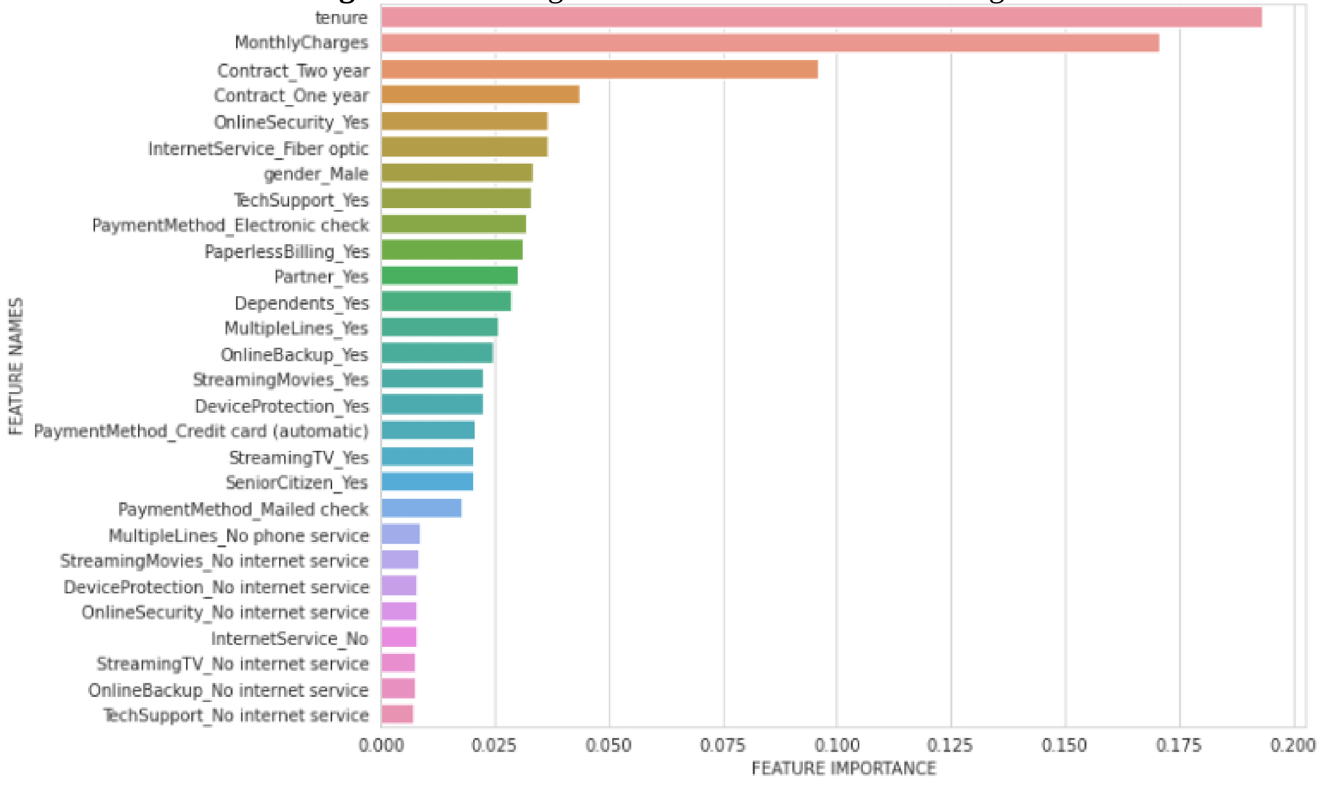


Figure B.5. – CatBoost One-Hot Featured Feature Weights

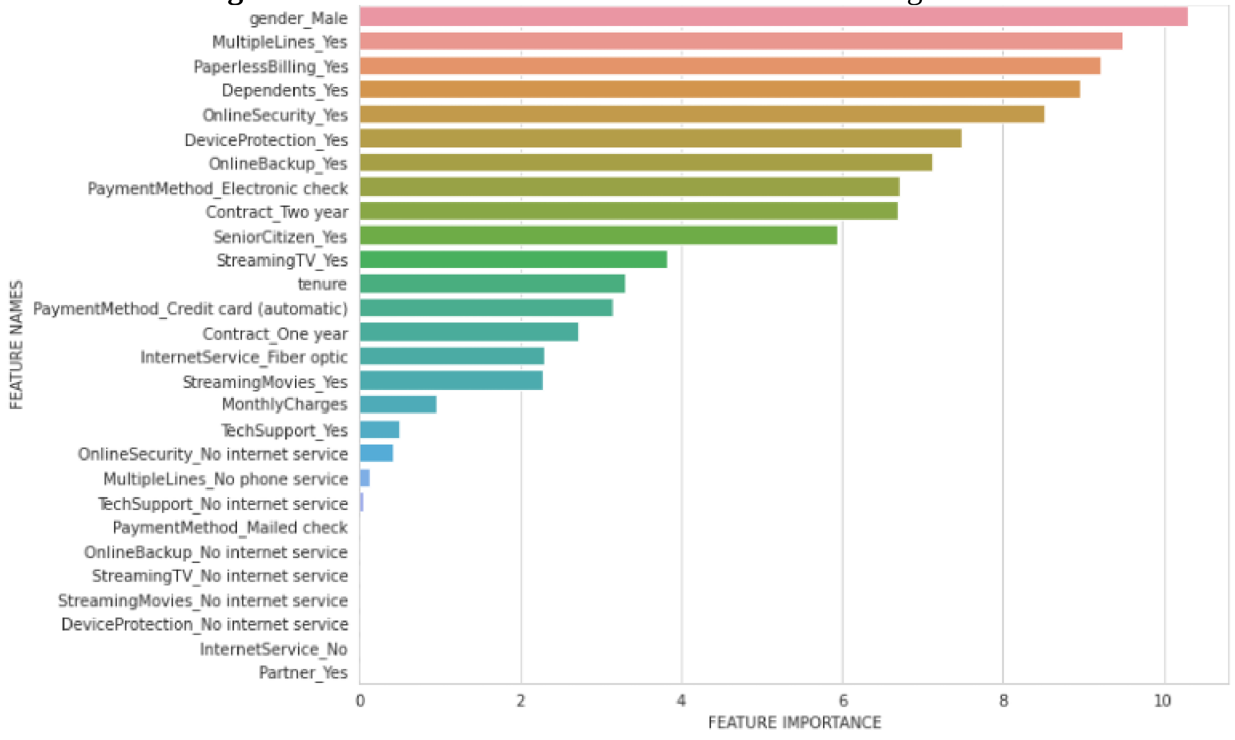
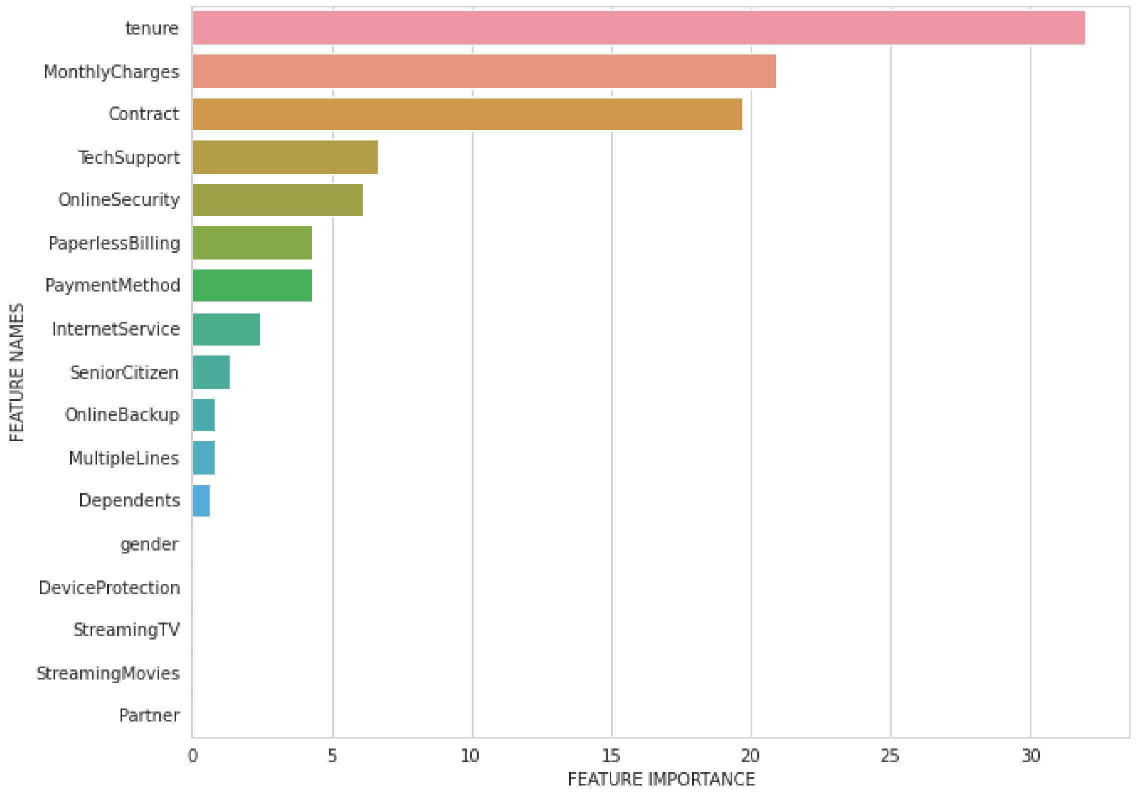


Figure B.6. – CatBoost Categorical (no one-hot) Features Feature Weights



C. Model Prediction Confusion Matrix

Figure C.1. – Logistic Regression (baseline) Accuracies

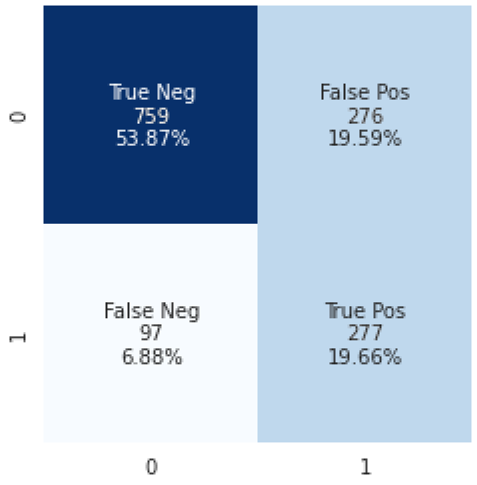


Figure C.2. – Random Forest Accuracies

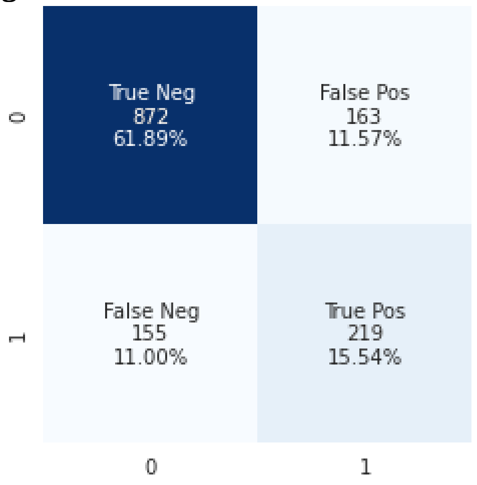


Figure C.3. – XGBoost Accuracies

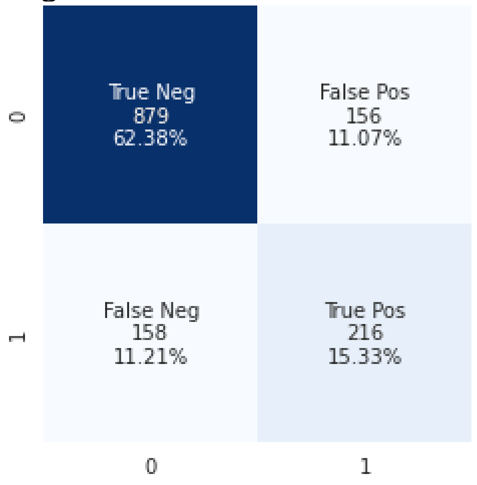


Figure C.4. – Support Vector Machine Accuracies

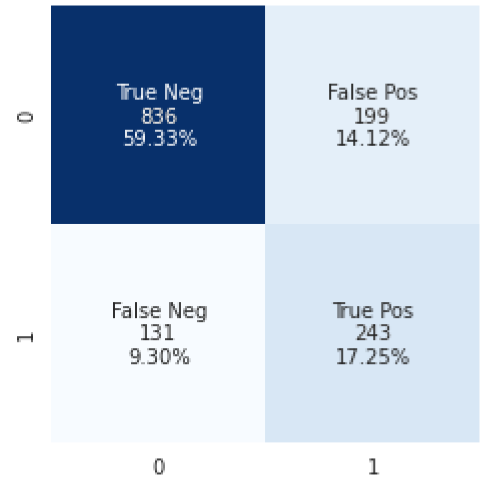


Figure C.5. – Weighted Random Forest Accuracies

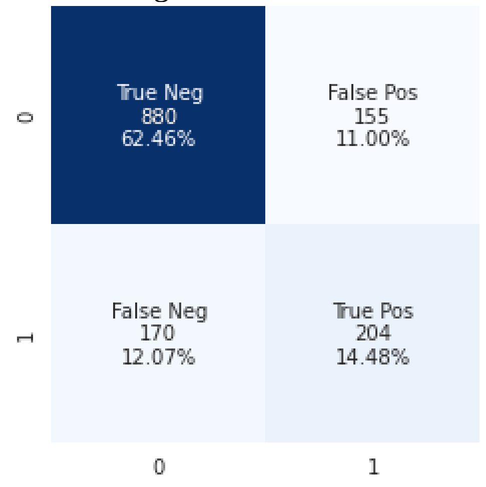


Figure C.6. – CatBoost Accuracies using one-hot encoded features

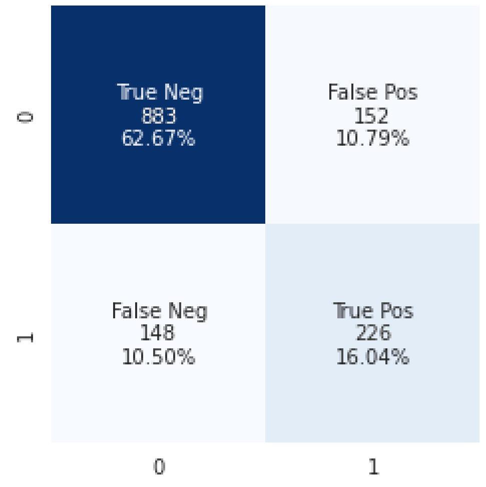
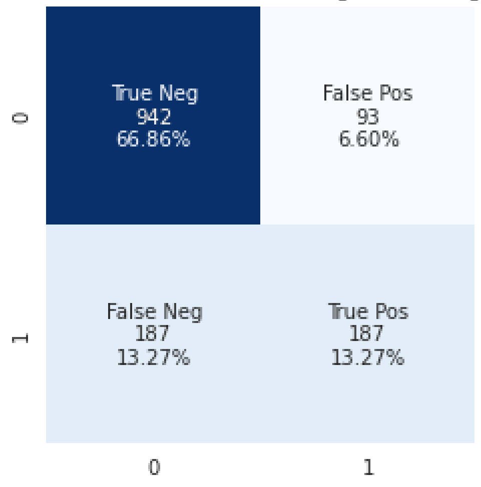


Figure C.7. - CatBoost Accuracies using raw categorical features


True Neg
False Pos
False Neg
True Pos