Andrew Mevissen
October 17th, 2018

Project Overview:

It is often the case that companies want to better understand their customers, for instance to be able to predict which customers are more likely to continue using their service and which will not. Churn is when a customer stops using a company's service. Being able to predict churn is useful to a company. It can be used to help predict the future number of customer various services will use, or it can help show which services may be lacking.

The goal is to determine whether or not the customer will churn based on the available data. The approach that will be used is supervised learning as we will look to learn the types of customers that are likely to stay and those that are likely to leave.

Past studies have compared various supervised learning approaches such as Caruana and Niculescu-Mizil's "An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics." They found that boosted trees, random forests, and unscaled neural nets were the best models [1]. Markham's "Comparing Supervised Learning Algorithms," compares the strengths and weakness of various supervised learning approaches [2]. Sabbeh's "Machine-Learning Techniques for Customer Retention: A comparative Study," was similar to Caruana and Niculescu-Mizil's in goal, but used telecom customer retention data. They also found that random forests and ADA boost had the highest accuracy at about 96%, and Multi-layer perceptrons and Support vector machines at 94% accuracy [3]. In a case study examining customer churn PayPal's "Solving Customer Churn with Machine Learning," used random forests [4].

Problem Statement:

This project will use two supervised learning approaches to predict customer churn. The first approach will be Scikit-Learn's Random Forest algorithm, and the other approach will be a custom neural network using Keras. These algorithms will be used to predict whether or not an individual customer continues to use the company's services.

Metrics:

The primary metrics that will be used to measure the performance of the models will be accuracy and F-score. The accuracy will be calculated according to Equation 1 [3], and the F-score will be calculated according to Equation 2 [5].

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN) \quad \text{Equation 1}$$

$$\text{F-Score} = (1+\beta^2)*TP/((1+\beta^2)*TP+ \beta^2*FN+FP) \quad \text{Equation 2}$$

Where TP is true positive, TN is true negative, FP is false positive and FP is false negative.

Accuracy is the percentage of customers that are correctly predicted to either turn or not churn out of the number of samples tested. The F-score (when β=0.5) is an average of both precision and recall. In this scenario, precision is the number of customers that are correctly predicted to have churned out of all customers predicted to churn, and recall is the number of customers that are correctly predicted to have churned out of all customers that actually churned [5]. While the F-score focuses on how well the model predicts the churn, the accuracy also factors in how well the model predicts those that do not churn.

Data Exploration and Visualization:

The data set was downloaded from Kaggle, but originates from an IBM sample data set. There is data on 7043 customers with 21 features (including whether or not the customer was retained). The features include a customer ID, the gender of the customer, whether or not the customer is a senior citizen, whether or not the customer has a partner, whether or not the customer has dependents, the number of months the customer has used the service (tenure), whether or not the customer has phone service, if the customer has phone service does the customer have more than one line, type of internet the customer has (if any), whether or not the customer has online security, whether or not the customer has an online backup, whether or not the customer has device protection, whether or not the customer has tech support, whether or not the customer has streaming TV, whether or not the customer streams movies, the type of contract, whether or not customer uses paperless billing, the type of billing the customer uses, the amount the customer pays each month, the total amount the customer has paid, and the whether or not the customer stayed (churn). A sample of the data can be seen in Figure 1.

customerID	gender	SeniorCitizen	Partner
7590-VHVEG	Female	0	Yes
Dependents	tenure	PhoneService	MultipleLines
No	1	No	No phone service
InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
DSL	No	Yes	No
TechSupport	StreamingTV	StreamingMovies	Contract
No	No	No	Month-to-month
PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
Yes	Electronic check	29.85	29.85
Churn			
No			

Figure 1. Data from a single customer.

Most of the data is categorical. The gender of the customer is given as either Male or Female. The Senior Citizen status is given as a 0 or a 1. The Partner status, Dependents status, Phone service status, Paperless Billing status, and Churn are given as either a Yes or a No. Multiple Lines status is given as Yes, No, or No phone service. Internet Service

status is given as DSL, Fiber optic, or No. Online Security status, Online Backup status, Device Protection status, Tech Support status, Streaming TV status, and Streaming Movies status are given as Yes, No, or No internet service. The Contract status is given as either Month-to-month, One year, or Two year. The Payment Method is given as Electronic check, Mailed check, Bank transfer (automatic), or Credit card (automatic).

Tenure, Monthly Charges, and Total Charges are non-categorical numerical values. Figure 2 gives minimum, maximum, range, mean, standard deviation, Quartile 1, Quartile 3, the lower range of Tukey's fences and the upper range of Tukey's fences. Tukey's fences is one method of determining whether or not a value is an outliner [6]. Using Tukey's fences for determining outliers, there does not appear to be any outliers.

	tenure	MonthlyCharges	TotalCharges
Minimum	0	118.75	8684.8
Maximum	72	18.25	18.8
Range	72	100.5	8666
Mean	32.37	64.76	2283.30
Standard Deviation	24.56	30.09	2266.77
Quartile 1	9.00	35.50	401.45
Quartile 3	55.00	89.85	3794.74
Tukey (low)	-60.00	-46.03	-4688.48
Tukey (high)	124.00	171.38	8884.67

Figure 2. Tenure, Monthly Charges, and Total Charges statistical summary.

For those customers with a tenure of 0, the Total Charges was cell was blank in the raw data set. Blank cell values were changed to 0 as they had not yet paid a monthly charge.

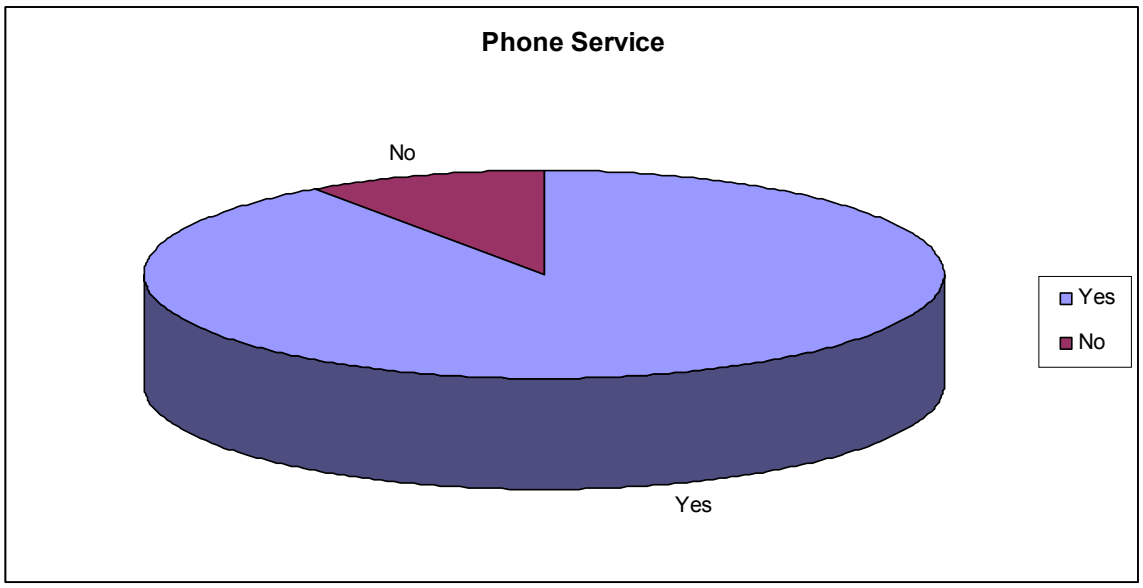

Figure 3. Proportion of customers with Phone Service vs. without Phone Service

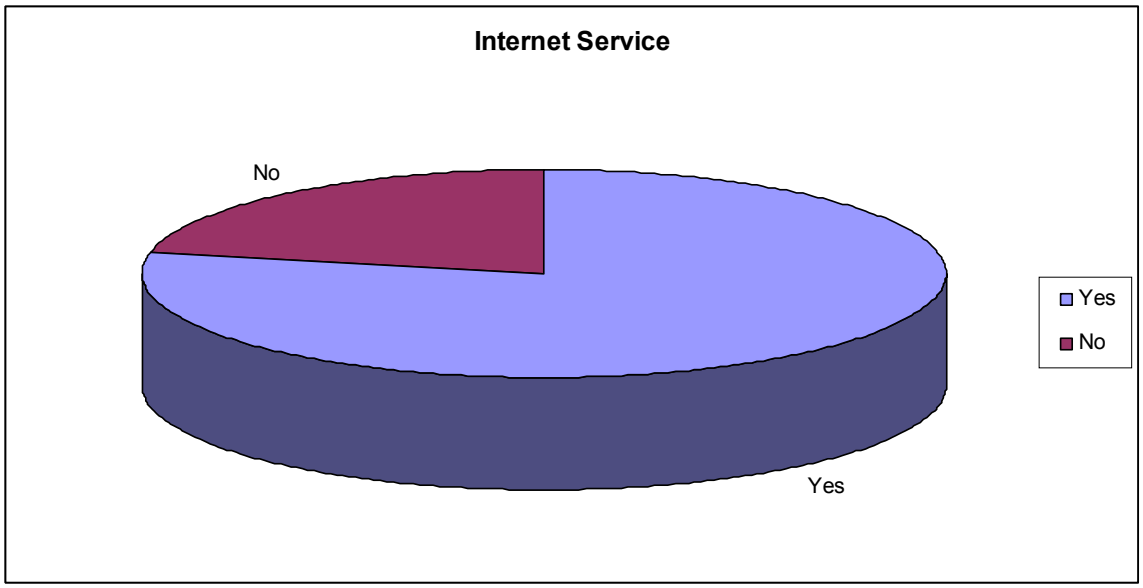

Figure 4. Proportion of customers with Internet Service vs. without Internet Service

Figures 3 and 4 show the proportion of customers with phone service vs. without and with internet service vs. without respectively. These specific features are important as other features are based on them. One can not have multiple phone lines if one does not have phone service. One can not have Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, and Streaming Movies if one does not have internet service.

Algorithms and Techniques:

As mentioned above the two approaches that will be used are Scikit-Learn's Random Forest algorithm a neural network using Keras.

In "An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics," random forests were amongst the highest performers with an accuracy ranging from 84.4% to 86.6% depending on if (and if so how), the data was calibrated [1]. In "Machine-Learning Techniques for Customer Retention: A comparative Study," random forests had an accuracy of about 96% [3]. In both studies random forests were amongst the highest accuracy.

In "Machine-Learning Techniques for Customer Retention: A comparative Study," Multi-layer perceptrons had an accuracy of about 94% [3]. This was also quite high.

Data used will also play a role in determining the accuracy. It will be easier to predict the results from some data compared to other data. For example if one set of data has more noise, its accuracy and F-score will be lower, through no fault of the algorithm used.

Benchmark:

The models will be compared to a guess. 73% of the people in the sample data remained and 27% churned. Guessing that the customer will remain will be correct 73% of the time; while guessing that the customer churned will be correct 27% of the time. This means that the most accurate guess is to guess that the customer will stay. Therefore, for the model to be useful it needs to produce results that are more accurate than guessing that the customer will remain.

Data Preprocessing:

The first step was to separate the features from the labels. This was done by making the Churn data its own variable. The Customer ID was also dropped from the data set as it is specific to that individual customer and not generalizable.

As previously mentioned, it was found that for those customers with a Tenure of 0 the Total Charges cell was empty. It was here that the empty cell values were changed to 0 as they had not yet paid a monthly charge.

The numerical data was normalized between zero and one. Due to the large range of the Total Charges (over 8000), the log of data values were found first. Then Tenure, Monthly Charges, and Total Charges were normalized using scikit-learn's MinMaxScaler function.

The rest of the features were categorical. The Senior Citizen did not need any preprocessing as it was already in terms of ones and zeros. The Gender, Partner, Dependents, Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies, Contract, Paperless Billing, and Payment Methods were one hot encoded using pandas get_dummies function. The labels (Churn data) were similarly one hot encoded.

The final preprocessing step was to divide data into a testing set and a training set for the random forest algorithm and testing set, validation set, and testing set for the neural network using scikit-learn's train_test_split function. For the neural network the data was converted from a data frame to numpy arrays.

Implementation:

To implement the random forest algorithm the following steps were taken:

1. Importing the classifier, the accuracy, and the f-score functions from sklearn
2. Creating the classifier
3. Fitting the training data using the classifier

4. The classifier is tested using the test data features to predict the outcome, these predicted labels are compared to the actual labels.

To implement the neural network the following steps were taken:

1. Importing the Dropout, Dense, Sequential, ModelCheckpoint, and EarlyStopping functions from Keras
2. Specify the size of the layers, the activation function for the layers, and the dropout probability of the layers.
3. Create the sequential network:
    a. The network is initialized
    b. The input layer is created, which includes the shape of the input data
    c. The first dropout layer is created
    d. A loop adds successive dense and dropout layers until the output layer is to be created
    e. The output layer is created
4. The network is compiled
5. The EarlyStopping and ModelCheckpoint callbacks are created
6. The model is trained using the training and validation data
7. The model uses the test data to predict the test label
    a. A model prediction variable is initialized
    b. The test data is walked through via a loop
    c. The value is formatted to work with Scikit Learn's accuracy and f-score functions
8. The predicted values are compared to the actual values

There were two primary changes from the originally planed approach to the final approach. The first was to convert the input data from a data frame to a numpy array to work with the neural network. The second changed was related to a step to preprocess the output of the neural network prediction to work with the accuracy and F-score functions. The accuracy and F-score functions needed integers not the probabilistic results that the neural net output. So the higher score was converted to a one and the lower score was converted to a zero.

Refinement:

To finalize the parameters of the random forest classifier a grid search was conducted utilizing a variety of n_estimators, max_depth, and min_sample_splits. Random forest is an ensemble method that makes a number of decision trees. N_estimators are the number decision trees used. Max_depth is the max depth of the decision trees. Min_sample_splits is the minimum number of samples to continue splitting [7]. The n_estimators used were: 1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, and 500. The max_depth values used were: None, 5, 7, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, and 500. The min_samples_split used were: 2, 5, 7, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, and 500.

The best values were 10 n_estimators, a max_depth of 'None' (no limit), and a min_sample_splits of 80. This combination had an accuracy of accuracy of 81.8% and an F-score of 0.665. The top 40 configurations are listed in Appendix I. All of the values can be found in the included spreadsheet titled RFScore_sorted.xls. This data set and algorithm does not seem to very sensitive to the values used.

The first configuration of the neural network that I tried was for the input layer to be 32 nodes, followed by a dropout layer with a 30% drop probability, followed by a hidden layer with 64 nodes and an rectified linear unit activation function, followed by a dropout layer with a 30% drop probability, followed by a hidden layer with 128 nodes and an rectified linear unit activation function, followed by a dropout layer with a 30% drop probability followed by the output layer with two nodes and a sigmoid activation function. That configuration had an accuracy of 79.7% and an F-score of 0.621. After the first run I tried to improve the neural network by varying the number of nodes in the layers, changing the number of hidden layers, changing the activation function, and the drop out rates. All of the runs are in Appendix II in the order that they were tried.

The best configuration was an input layer with 128 nodes followed by a dropout layer with a 30% dropout probability, followed by a hidden layer with 256 nodes and an exponential linear function activation followed by a second dropout with 30% probability followed by an output layer with 2 nodes and a sigmoid activation function. This neural network had the best single run accuracy of 81.4% and f-score of 0.657.

Model Evaluation, Validation, and Justification:

Figure 5 shows the final Random Forest Model and Figure 6 shows the final neural network model.

```
#Running the Random Forest (Final Model) :
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, fbeta_score, make_scorer
from sklearn import grid_search
import pickle

#Model parameters
n_estimators=10
max_depth=None
min_samples_split=80

#Classifier
RandomForest=RandomForestClassifier(n_estimators=n_estimators,
                                    max_depth=max_depth, min_samples_split=min_samples_split)

#Training the Model
RandomForest.fit(X_train_RF, y_train_RF['Yes'])

print(n_estimators, max_depth, min_samples_split,
      accuracy_score(y_test_RF['Yes'], RandomForest.predict(X_test_RF)),
      fbeta_score(y_test_RF['Yes'], RandomForest.predict(X_test_RF), 0.5))

#Saving the Model:
RandomForestfile='RandomForestModel.sav'
pickle.dump(RandomForest, open(RandomForestfile, 'wb'))
```

Figure 5. Random Forest Final Model

```
: #neural network:
from keras.layers import Dropout, Dense
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import accuracy_score, fbeta_score


#Network Shape Inputs:
size=[128, 256, 2]
act=['', 'elu','sigmoid']
drop=[0.3, 0.3]

#Building the Network:
Prime_model = Sequential()
Prime_model.add(Dense(size[0], input_shape=(X_train_NN.shape[1],)))
Prime_model.add(Dropout(drop[0]))

for i in range(len(size)-2):
    Prime_model.add(Dense(size[i+1], activation=act[i+1]))
    Prime_model.add(Dropout(drop[i+1]))

Prime_model.add(Dense(size[-1], activation=act[-1]))
#Prime_model.summary()


#compiling the Network:
Prime_model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

#Training the Network:
epochs = 30

Stopping = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=1, mode='auto')
Saving=ModelCheckpoint(filepath='G:\Machine Learning\weightsdense.best.from_scratch.hdf5',
                       monitor='val_loss', verbose=1, save_best_only=True)


Prime_model.fit(X_trainNP, y_trainNP,
          validation_data=(X_valNP, y_valNP),
          epochs=epochs, batch_size=7, callbacks=[Stopping, Saving])



#Evaluating the results:
Prime_model.load_weights('G:\Machine Learning\weightsdense.best.from_scratch.hdf5')

y_pred=np.zeros((len(X_testNP),1))

for i in range(len(X_testNP)):
    temp=Prime_model.predict(np.array([X_testNP[i]]))
    temp+=0.5
    if temp[0][0]>temp[0][1]:
        y_pred[i]=0
    else:
        y_pred[i]=1


print ('accuracy: ', accuracy_score(y_test_NN['Yes'], y_pred))
print ('f-score: ', fbeta_score(y_test_NN['Yes'], y_pred, 0.5))
```

Figure 6. Final Neural Network Final Model

The final models were chosen because they had both the highest accuracy and F-scores of their respective methods. The random forest had and accuracy of 81.8% and an F-score of 0.665. The neural network had and accuracy of 81.4% and an F-score of 0.657.

As can be seen in Figure 7, reducing the amount of data slowly decreases the accuracy and F-scores of the data. Using 1% of the data (about 70 samples) the random forest model isn't better than guessing that all of the customers will stay (the benchmark); while

the neural network is only more accurate about 4% of the time. Even at their best the Random forest and neural network do not look much better than guessing that every customer will stay (approximately 81% vs 73%). It is in the F-score that both models shine.

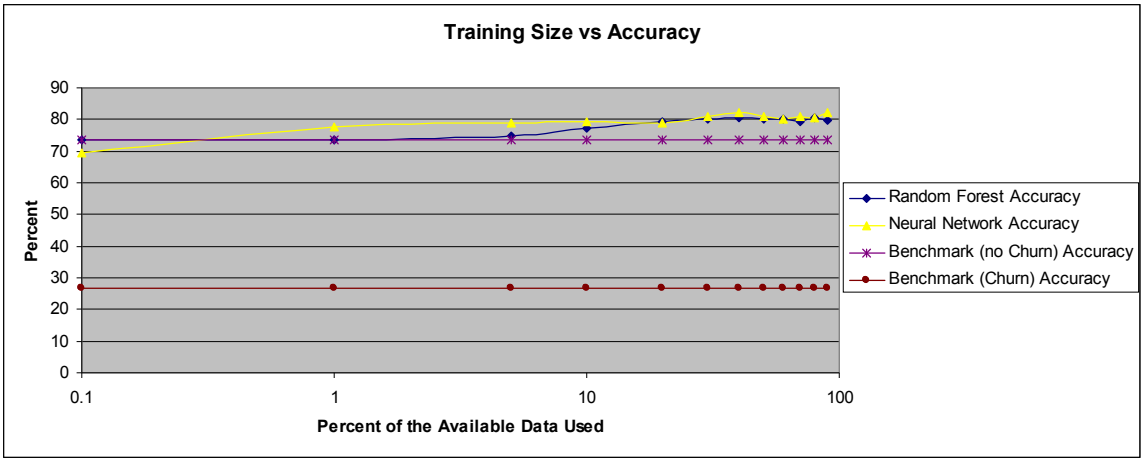

Figure 7. Training Size vs. Accuracy

If one assumes that every customer stays (which the majority of them do), you will never catch the customers that churn. Guessing that the customers stay has an F-score of 0. This is where the models are significantly better. As can be seen in Figure 8 both models have F-scores of in the low 60s. This trend continues until using about 20% of the data for training (about 1400 samples). Using less than 20% of the available data, the random forest's F-score starts to precipitously drop. It is likely, based on its accuracy in this range, that it is guessing that most of the customers do not churn (the same as the benchmark). While the neural network does not drop as much in this range, using less than 1% of the data, the accuracy drops below the accuracy benchmark. This suggest that it is erring on the side of customer churn.

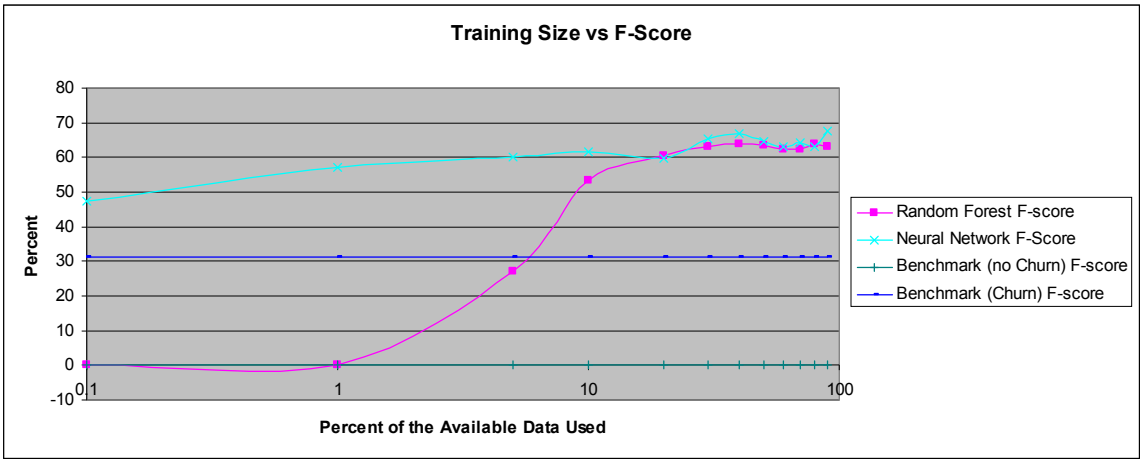

Figure 8. Training Size vs. F-score

While neither method was able to correctly predict the whether a customer would churn or not 100% of the time, both methods significantly improve upon the benchmark scores, specifically the F-scores.

Reflection and Free-Form Visualization:

The basic flow of the problem was a follows:

1. Downloading and importing the dataset
2. Preprocessing the data by normalizing the non-categorical numerical features and one hot encoding the categorical features and labels.
3. Separating the data into train and test data for the random forest and train, validate, and test data for the neural network.
4. Exploring the parameter space of the models
    a. Random Forest:
        i. Creating the classifier
        ii. Fitting the classifier
        iii. Measuring the Accuracy and F-score
        iv. Repeating for a variety of different parameters
    b. Neural Network:
        i. Building the network
        ii. Compiling the network
        iii. Training the network
        iv. Measuring the Accuracy and F-score
        v. Repeating for a variety of different parameters
5. Using the best combination of parameters to set the final models
6. Testing the robustness of the solutions by changing the amount of training data the models received.

I was surprised at how similar both the accuracy and F-score were for both models. Additionally I was surprised at how little effect perturbing the model parameters had. For the neural network I tried adding additional hidden layer, removing hidden layers, changing the number of nodes in the input and hidden layers, having the number of nodes increase with each layer, decrease with each layer, tried different activation functions and most of the models in the ranges that I tried gave similar results.

The same can be seen in the Random Forest parameters. Looking at the 40 best results in Appendix I there is very little change in the accuracy and F-score between the best set of parameters and the 40^{th}. However, looking at all of the data some noticable trends do form. Figures 9 and 10 plot how changing the parameters changes the accuracy and F-scores. For instance data point of the n_estimators line averages accuracy and F-score for all runs with an e_estimator of 1. This is done for both the max_depth and min_samples_split as well. On average, having fewer than about 20 estimators seems detrimental; however, little, if anything, is gained from having much more than 20 estimators. Having a max_depth of 7 is, on average the best; however, the effect is small (about 0.5% on accuracy and 0.015 on F-score). The min_samples_split has more

variation. If the there min number of samples to split on is too low the accuracy and F-score decrease, this could be a sign of over fitting. However, if the value is too large accuracy and F-score also decrease, this could be a sign of under fitting.

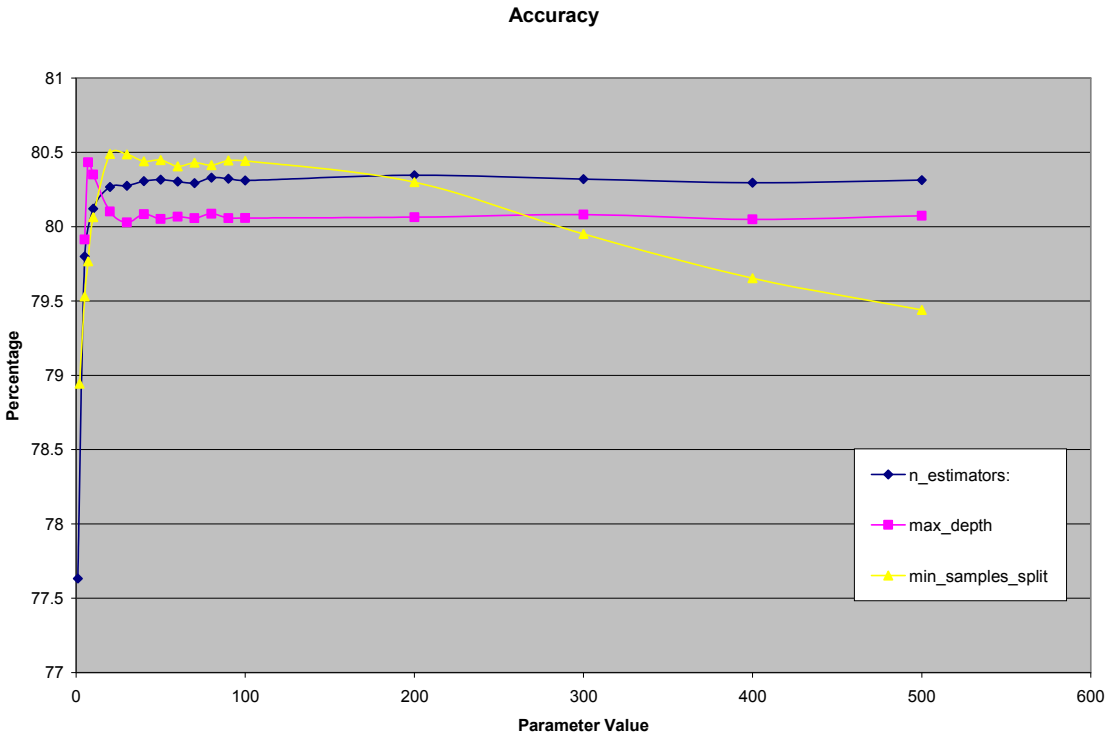

Figure 9. Random Forest Accuracy vs. Parameter Value

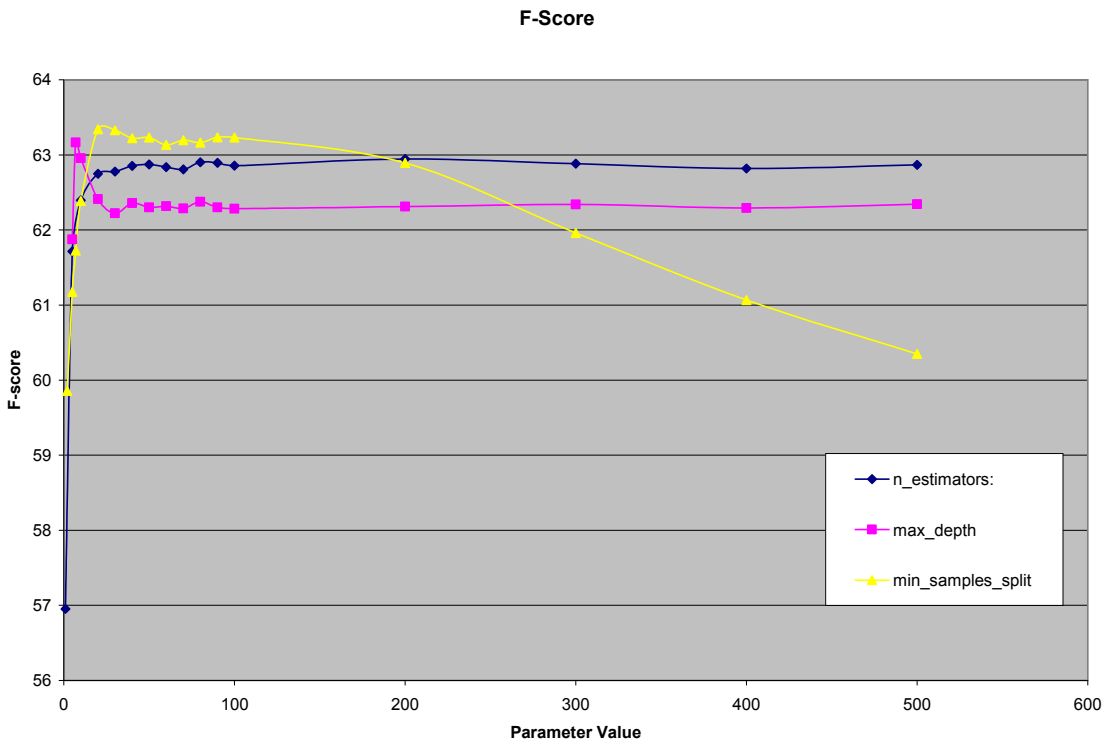

Figure 10. Random Forest F-score vs. Parameter Value

Improvement:

It was noticed that there are run to run variations. Rerunning the model parameters multiple times and averaging the results (resplitting the data into train, (validate), and train each time) may lead to a different set of parameters being better on average. However, judging from the lack of change in results from perturbing the model parameters, it is unlikely to improve by more than a percent or two at best.

A more useful change to the model would be to gain a deeper insight into the data. Scikit Learn's Random Forest algorithm has built in functions to help isolate which features are the most significant. This may be useful in finding the scenarios where people were the most likely to leave. With this data the company would know which services to improve that would most likely have the biggest effect on customer retention.

Sources:

[1] Caruana, Rich, and Alexandru Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics." Proceedings of ICML'06 (n.d.): n. page. Web. http://www.niculescu-mizil.org/papers/comparison.tr.pdf

[2] Markham, Kevin. "Comparing Supervised Learning Algorithms." Data School. N.p., 25 Feb. 2015. Web. http://www.dataschool.io/comparing-supervised-learning-algorithms/.

[3] Sabbeh, Sahar. "Machine-Learning Techniques for Customer Retention: A comparative Study." International Journal of Advanced Computer Science and Applications Vol. 9, No 2, 2018: Web. http://thesai.org/Downloads/Volume9No2/Paper_38-Machine_Learning_Techniques_for_Customer_Retention.pdf

[4] "Solving Customer Churn with Machine Learning." PayPal. https://www.h2o.ai/wp-content/uploads/2017/03/Case-Studies_PayPal.pdf

[5] "F1 Score." Wikipedia. 29 September 2018. https://en.wikipedia.org/wiki/F1_score

[6] "Outlier." Wikipedia. 10 September 2018. https://en.wikipedia.org/wiki/Outlier

[7] "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier." Scikit Learn. 15 October 2018. **http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html**

Data:

The data was originally found on Kaggle:
https://www.kaggle.com/blastchar/telco-customer-churn

The original source of the data is IBM:
https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/

Appendix I

Random Forest Grid Search (top samples):

n_estimators	max_depth	min_samples_split	Accuracy	F-score
10	nan	80	81.76%	66.48%
10	nan	20	81.83%	66.38%
10	100	100	81.62%	66.21%
5	200	70	81.62%	66.11%
40	10	7	81.76%	66.10%
5	200	40	81.62%	65.96%
30	20	30	81.55%	65.95%
5	10	40	81.62%	65.90%
10	100	70	81.55%	65.89%
20	70	30	81.62%	65.88%
5	60	30	81.55%	65.87%
5	70	100	81.48%	65.84%
20	7	10	81.55%	65.82%
20	200	30	81.55%	65.82%
10	30	20	81.55%	65.73%
20	40	50	81.48%	65.71%
90	70	20	81.48%	65.66%
200	80	20	81.48%	65.66%
20	80	40	81.48%	65.62%
5	300	90	81.41%	65.56%
80	500	30	81.41%	65.55%
60	30	30	81.41%	65.53%
10	400	40	81.41%	65.51%
20	500	40	81.41%	65.50%
60	10	10	81.48%	65.50%
5	500	100	81.41%	65.46%
50	7	60	81.41%	65.46%
50	300	20	81.41%	65.46%
5	30	40	81.41%	65.45%
20	7	20	81.41%	65.43%
20	20	7	81.33%	65.38%
60	60	40	81.33%	65.37%
60	90	30	81.33%	65.35%
70	90	40	81.33%	65.35%
80	100	20	81.33%	65.35%
20	20	60	81.33%	65.34%
70	70	10	81.33%	65.34%
50	40	20	81.33%	65.32%
5	20	60	81.33%	65.29%
5	200	60	81.33%	65.28%

Appendix II

Neural Network Search:

Size	Act	Drop	Accuracy	F-score
[32, 64, 128, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	79.70%	0.62137
[64, 128, 256, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	80.91%	0.652174
[128, 256, 512, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	79.91%	0.625948
[128, 256, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	80.34%	0.636238
[64, 128, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	80.84%	0.646067
[128, 256, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	80.34%	0.636115
[96, 192, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	80.84%	0.649073
[32, 64, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	80.41%	0.636763
[64, 128, 256, 512, 2]	['','relu','relu','relu', 'sigmoid']	[0.3, 0.3, 0.3, 0.3]	80.34%	0.635935
[32, 64, 128, 256, 2]	['','relu','relu','relu', 'sigmoid']	[0.3, 0.3, 0.3, 0.3]	79.63%	0.619497
[16, 32, 64, 128, 2]	['','relu','relu','relu', 'sigmoid']	[0.3, 0.3, 0.3, 0.3]	78.71%	0.59417
[48, 96, 192, 384, 2]	['','relu','relu','relu', 'sigmoid']	[0.3, 0.3, 0.3, 0.3]	79.70%	0.620783
[64, 128, 256, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	80.06%	0.62989
[64, 128, 32, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	80.13%	0.630462
[32, 64, 32, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	80.20%	0.633484
[32, 64, 16, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	79.77%	0.622677
[64, 32, 16, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	79.56%	0.616225
[32, 16, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	80.62%	0.640871
[64, 128, 256, 2]	['','relu', 'relu', 'sigmoid']	[0.2, 0.2, 0.2]	80.55%	0.641201
[64, 128, 256, 2]	['','relu', 'relu', 'sigmoid']	[0.4, 0.4, 0.4]	80.48%	0.638432
[64, 128, 256, 2]	['','relu', 'relu', 'sigmoid']	[0.3, 0.3, 0.3]	80.62%	0.646481
[48, 96, 2]	['','relu', 'sigmoid']	[0.3, 0.3]	79.42%	0.614323
[64, 128, 256, 2]	['','tanh','tanh', 'sigmoid']	[0.3, 0.3, 0.3]	80.55%	0.640362
[32, 64, 128, 256, 2]	['','tanh','tanh', 'tanh','sigmoid']	[0.3, 0.3, 0.3, 0.3]	80.20%	0.631835
[32, 64, 128, 256, 2]	['','elu','elu', 'elu', 'sigmoid']	[0.3, 0.3, 0.3, 0.3]	80.48%	0.638232
[64, 128, 256, 2]	['','elu', 'elu', 'sigmoid']	[0.3, 0.3, 0.3]	80.91%	0.647239
[128, 256, 512, 2]	['','elu', 'elu', 'sigmoid']	[0.3, 0.3, 0.3]	80.84%	0.646388
[32, 64, 128, 2]	['','elu', 'elu', 'sigmoid']	[0.3, 0.3, 0.3]	78.85%	0.60582
[96, 192, 384, 2]	['','elu', 'elu', 'sigmoid']	[0.3, 0.3, 0.3]	80.48%	0.637848
[48, 96, 192, 2]	['','elu', 'elu', 'sigmoid']	[0.3, 0.3, 0.3]	80.48%	0.637312
[64, 128, 2]	['', 'elu','sigmoid']	[0.3, 0.3]	80.27%	0.63367
[128, 256, 2]	['', 'elu','sigmoid']	[0.3, 0.3]	81.41%	0.657263
[256, 512, 2]	['', 'elu','sigmoid']	[0.3, 0.3]	79.91%	0.625752
[192, 384, 2]	['', 'elu','sigmoid']	[0.3, 0.3]	79.77%	0.622744
[128, 256, 2]	['', 'elu','sigmoid']	[0.2, 0.2]	81.26%	0.65331
[128, 256, 2]	['', 'elu','sigmoid']	[0.4, 0.4]	79.70%	0.620872
[128, 256, 2]	['', 'elu','sigmoid']	[0.3, 0.3]	80.27%	0.634034