Customer Tenure Analysis Using Random Forest Regression

Douglas Ehlert

Customer Tenure Analysis Using Random Forest Regression

Part 1: Research Question

To maintain profits, it is critical for a telecommunication organization to retain customers for as long as possible. This helps to offset customer acquisition costs and reduced the spend needed on sales and marketing. This report will examine the effects of numerous variables on customer tenure. The research question is: which variables have the highest effect on customer tenure? The goal of the analysis is to decide whether a model can be created to accurately predict customer tenure. This would enable the organization to target and interact with customers with the knowledge of what variables affect customer tenure.

Part II: Method Justification

Random Forest (RF) a simple, yet powerful, tool used to perform prediction. RF regression is an Ensemble Learning method with a decision tree as its base. RF aggregates predictions through averaging and is useful with a large amount of independent (input) variables. RF is "designed to decrease overfitting and variance by using bagging algorithms" (Lorberfeld, 2019).

The model will attempt to predict customer tenure, churn, by analyzing the training data set to make predictions through averaging. One would expect that if the independent variables in the training dataset have indicated a given customer tenure, similar independent variables will lead to similar churn results in the test dataset. List of imported libraries and packages:

- Pandas
    - Used for data manipulation such as import and of export of .csv files, data wrangling, analysis, and cleaning

- Label Encoder/OneHotEncoder
    - Used to prep data. (Converting non-integer categorical variables to integers)
- RandomForestRegressor
    - Used to instantiate the model for analysis
- Train_test_split
    - Used to split the dataset into training and test data
- GridSearchCV
    - Used to find the ideal model parameters
- Mean_squared_error
    - Used to find the RMSE for model evaluation

Part III: Data Preparation

The initial data preparation goal is to import the dataset and libraries/packages, perform an overview of the data (columns, datatype, etc.), and discover any missing values. The following four pictures show these steps in the code (Jupyter Notebook).

In [1]:
```
#import libraries

import pandas as pd

#import encoder to deal with categorical variables
#import other libraries needed for sklearn module for data prep, kNN m
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import GridSearchCV
```

In [2]:
```
df = pd.read_csv('/Users/dougehlert/Desktop/D209-2/churn_clean.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	CaseOrder	Customer_id	Interaction	UID	City	Stat
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	Al
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	M
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OI
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	C/
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	T/

5 rows × 50 columns

In [4]:
```
dataset = df.drop(['Customer_id', 'Interaction', 'CaseOrder', 'Job','T
```

In [5]:
```
#This is a SUMMARY STATISTIC of the dataset.
#It shows various metrics for ALL the predictors.
dataset.describe()
```

Out[5]:

	Children	Age	Income	Outage_sec_perweek	Email	Cont
count	10000.0000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000
mean	2.0877	53.078400	39806.926771	10.001848	12.016000	0.994
std	2.1472	20.698882	28199.916702	2.976019	3.025898	0.988
min	0.0000	18.000000	348.670000	0.099747	1.000000	0.000
25%	0.0000	35.000000	19224.717500	8.018214	10.000000	0.000
50%	1.0000	53.000000	33170.605000	10.018560	12.000000	1.000
75%	3.0000	71.000000	53246.170000	11.969485	14.000000	2.000
max	10.0000	89.000000	258900.700000	21.207230	23.000000	7.000

In [6]:
```
#Check data for data types.
dataset.dtypes
```

Out[6]:
```
Area                   object
Children                int64
Age                     int64
Income                float64
Marital                object
Gender                 object
Churn                  object
Outage_sec_perweek    float64
Email                   int64
Contacts                int64
Yearly_equip_failure    int64
Techie                 object
Contract               object
Port_modem             object
Tablet                 object
InternetService        object
Phone                  object
Multiple               object
OnlineSecurity         object
OnlineBackup           object
DeviceProtection       object
TechSupport            object
StreamingTV            object
StreamingMovies        object
PaperlessBilling       object
PaymentMethod          object
Tenure                float64
MonthlyCharge         float64
Bandwidth_GB_Year     float64
Item1                   int64
Item2                   int64
Item3                   int64
Item4                   int64
Item5                   int64
Item6                   int64
Item7                   int64
Item8                   int64
dtype: object
```

In [7]:
```
#confirm there are no null values
dataset.isna().any()
```

Out[7]:
```
Area                   False
Children               False
Age                    False
Income                 False
Marital                False
Gender                 False
Churn                  False
Outage_sec_perweek     False
Email                  False
Contacts               False
Yearly_equip_failure   False
Techie                 False
Contract               False
Port_modem             False
Tablet                 False
InternetService        False
Phone                  False
Multiple               False
OnlineSecurity         False
OnlineBackup           False
DeviceProtection       False
TechSupport            False
StreamingTV            False
StreamingMovies        False
PaperlessBilling       False
PaymentMethod          False
Tenure                 False
MonthlyCharge          False
Bandwidth_GB_Year      False
Item1                  False
Item2                  False
Item3                  False
Item4                  False
Item5                  False
Item6                  False
Item7                  False
Item8                  False
dtype: bool
```

Before analysis proceeds, dummy variables had to be created for the categorical variables, that are going to be used in the model, that were not integers. This is necessary for analysis as a Random Forest model would have no way of dealing with variables that consisted of different strings of text. Then the dummy variables are concatenated into the dataset. The target variable (dependent variable) is Tenure. The predictor variables (independent variables) are, initially, Area, Children, Interaction, Job, Age, Income, Marital, Gender, Email, Outage_sec_perweek, Contacts, Yearly_equip_failure, Techie, Contract, Port_modem, Tablet, InternetService, Phone, Multiple, OnlineSecurity, OnlineBackup, TechSupport, StreamingTV, StreamingMovies, PaperlessBilling, PaymentMethod, Churn, MonthlyCharge, Bandwidth_GB_Year, Item1, Item2, Item3, Item4, Item5, Item6, Item7, and Item8. Numerous variables were removed from the initial dataset because they were not pertinent to the analysis at hand. (CaseOrder, TimeZone, Customer_id, Job, Interaction, UID, City, State, County, Zip, Lat,Lng, Population). City, State, County, Zip, Lat, Lng, and Population were removed due existence of the variable Area. These variables would not be considered independent of Area.

The dependent variable, Tenure, is assigned to ‘y’ and the independent variables are assigned to ‘X’. The shape of both ‘X’ and ‘y’ is checked to ensure that the length is equal. The cleaned, concatenated dataset, with dummy variables is exported as “ChurnClean2.csv”. Independent, continuous variables include Children, Age, Income, Outage_sec_perweek, Email, Contacts, Yearly_equip_failure, MonthlyCharge, and Bandwidth_GB_Year. Independent, categorical variables include Techie, Port_modem, Tablet, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, PaperlessBilling, Item 1, Item 2, Item 3, Item 4, Item 5, Item 6, Item 7, Item 8, Rural, Suburban,

Urban, Female, Male, Nonbinary, Divorced, Married, Never Married, Separated, Widowed, DSL, Fiber Optic, None, BankTransfer(automatic), CreditCard(automatic), Electronic Check, Mailed Check, Month-to-month, One year, two year, and churn. The dependent variable, which must be continuous for regression models, is Tenure.

The following pictures show these steps, in the code, from Jupyter Notebook.

In [8]:
```
dataset["Tenure"].value_counts()
```

Out[8]:
```
69.504800    2
62.865710    2
66.668530    2
55.449910    2
2.021619     1
            ..
4.313377     1
61.000600    1
67.919360    1
7.819784     1
50.554520    1
Name: Tenure, Length: 9996, dtype: int64
```

In [9]:
```
#label encoder object
le = LabelEncoder()
# Columns with one or two values can have label encoder run on them.
le_count = 0
for col in dataset.columns[1:]:
    if dataset[col].dtype == 'object':
        if len(list(dataset[col].unique())) <= 2:
            le.fit(dataset[col])
            dataset[col] = le.transform(dataset[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
```

```
13 columns were label encoded.
```

In [10]:
```
# Creating dummy variable for Gender
Gender_cat = pd.get_dummies(dataset['Gender'])

# Check what the dataset 'status' looks like
Gender_cat
```

Out[10]:

	Female	Male	Nonbinary
0	0	1	0
1	1	0	0
2	1	0	0
3	0	1	0
4	0	1	0
...	...	...	...
9995	0	1	0
9996	0	1	0
9997	1	0	0
9998	0	1	0
9999	0	1	0

10000 rows × 3 columns

In [11]:
```
#getdummies will be used to assign values and create new columns to th
# Creating dummy variable for Area
Area_cat = pd.get_dummies(dataset['Area'])

# Check what the dataset 'status' looks like
Area_cat
```

Out[11]:

	Rural	Suburban	Urban
0	0	0	1
1	0	0	1
2	0	0	1
3	0	1	0
4	0	1	0
...	...	...	...
9995	1	0	0
9996	1	0	0
9997	1	0	0
9998	0	0	1
9999	0	0	1

10000 rows × 3 columns

In [12]:
```
# Creating dummy variable for Marital
Marital_cat = pd.get_dummies(dataset['Marital'])

# Check what the dataset 'status' looks like
Marital_cat
```

Out[12]:

	Divorced	Married	Never Married	Separated	Widowed
0	0	0	0	0	1
1	0	1	0	0	0
2	0	0	0	0	1
3	0	1	0	0	0
4	0	0	0	1	0
...	...	...	...	...	...
9995	0	1	0	0	0
9996	1	0	0	0	0
9997	0	0	1	0	0
9998	0	0	0	1	0
9999	0	0	1	0	0

10000 rows × 5 columns

In [13]:
```
# Creating dummy variable for InternetService
InternetService_cat = pd.get_dummies(dataset['InternetService'])

# Check what the dataset looks like
InternetService_cat
```

Out[13]:

	DSL	Fiber Optic	None
0	0	1	0
1	0	1	0
2	1	0	0
3	1	0	0
4	0	1	0
...	...	...	...
9995	1	0	0
9996	0	1	0
9997	0	1	0
9998	0	1	0
9999	0	1	0

10000 rows × 3 columns

In [14]:
```
# Creating dummy variable for PaymentMethod
PaymentMethod_cat = pd.get_dummies(dataset['PaymentMethod'])

# Check what the dataset 'status' looks like
PaymentMethod_cat
```

Out[14]:

	Bank Transfer(automatic)	Credit Card (automatic)	Electronic Check	Mailed Check
0	0	1	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	0	1
4	0	0	0	1
...	...	...	...	...
9995	0	0	1	0
9996	0	0	1	0
9997	1	0	0	0
9998	0	1	0	0
9999	0	0	1	0

10000 rows × 4 columns

In [15]:
```
# Creating dummy variable for Contract
Contract_cat = pd.get_dummies(dataset['Contract'])

# Check what the dataset 'status' looks like
Contract_cat
```

Out[15]:

	Month-to-month	One year	Two Year
0	0	1	0
1	1	0	0
2	0	0	1
3	0	0	1
4	1	0	0
...	...	...	...
9995	1	0	0
9996	0	0	1
9997	1	0	0
9998	0	0	1
9999	1	0	0

10000 rows × 3 columns

In [16]:
```
#Combining the dummy variables into the dataset
dataset = pd.concat([dataset, Area_cat], axis = 1)
dataset
```

Out[16]:

	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perweek	Em
0	Urban	0	68	28561.99	Widowed	Male	0	7.978323	
1	Urban	1	27	21704.77	Married	Female	1	11.699080	
2	Urban	4	50	9609.57	Widowed	Female	0	10.752800	
3	Suburban	1	48	18925.23	Married	Male	0	14.913540	
4	Suburban	0	83	40074.19	Separated	Male	1	8.147417	
...	...	...	...	...	...	...	...	...	
9995	Rural	3	23	55723.74	Married	Male	0	9.415935	
9996	Rural	4	48	34129.34	Divorced	Male	0	6.740547	
9997	Rural	1	48	45983.43	Never Married	Female	0	6.590911	
9998	Urban	1	39	16667.58	Separated	Male	0	12.071910	
9999	Urban	1	28	9020.92	Never Married	Male	0	11.754720	

10000 rows × 40 columns

In [17]:
```
#Combining the dummy variables into the dataset
dataset = pd.concat([dataset, Gender_cat], axis = 1)
dataset
```

Out[17]:

	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perweek	Em
0	Urban	0	68	28561.99	Widowed	Male	0	7.978323	
1	Urban	1	27	21704.77	Married	Female	1	11.699080	
2	Urban	4	50	9609.57	Widowed	Female	0	10.752800	
3	Suburban	1	48	18925.23	Married	Male	0	14.913540	
4	Suburban	0	83	40074.19	Separated	Male	1	8.147417	
...	...	...	...	...	...	...	...	...	
9995	Rural	3	23	55723.74	Married	Male	0	9.415935	
9996	Rural	4	48	34129.34	Divorced	Male	0	6.740547	
9997	Rural	1	48	45983.43	Never Married	Female	0	6.590911	
9998	Urban	1	39	16667.58	Separated	Male	0	12.071910	
9999	Urban	1	28	9020.92	Never Married	Male	0	11.754720	

10000 rows × 43 columns

In [18]:
```
#Combining the dummy variables into the dataset
dataset = pd.concat([dataset, Marital_cat], axis = 1)
dataset
```

Out[18]:

	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perweek	Em
0	Urban	0	68	28561.99	Widowed	Male	0	7.978323	
1	Urban	1	27	21704.77	Married	Female	1	11.699080	
2	Urban	4	50	9609.57	Widowed	Female	0	10.752800	
3	Suburban	1	48	18925.23	Married	Male	0	14.913540	
4	Suburban	0	83	40074.19	Separated	Male	1	8.147417	
...	...	...	...	...	...	...	...	...	
9995	Rural	3	23	55723.74	Married	Male	0	9.415935	
9996	Rural	4	48	34129.34	Divorced	Male	0	6.740547	
9997	Rural	1	48	45983.43	Never Married	Female	0	6.590911	
9998	Urban	1	39	16667.58	Separated	Male	0	12.071910	
9999	Urban	1	28	9020.92	Never Married	Male	0	11.754720	

10000 rows × 48 columns

In [19]:
```
#Combining the dummy variables into the dataset
dataset = pd.concat([dataset, InternetService_cat], axis = 1)
dataset
```

Out[19]:

	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perweek	Em
0	Urban	0	68	28561.99	Widowed	Male	0	7.978323	
1	Urban	1	27	21704.77	Married	Female	1	11.699080	
2	Urban	4	50	9609.57	Widowed	Female	0	10.752800	
3	Suburban	1	48	18925.23	Married	Male	0	14.913540	
4	Suburban	0	83	40074.19	Separated	Male	1	8.147417	
...	...	...	...	...	...	...	...	...	
9995	Rural	3	23	55723.74	Married	Male	0	9.415935	
9996	Rural	4	48	34129.34	Divorced	Male	0	6.740547	
9997	Rural	1	48	45983.43	Never Married	Female	0	6.590911	
9998	Urban	1	39	16667.58	Separated	Male	0	12.071910	
9999	Urban	1	28	9020.92	Never Married	Male	0	11.754720	

10000 rows × 51 columns

In [20]:
```
#Combining the dummy variables into the dataset
dataset = pd.concat([dataset, PaymentMethod_cat], axis = 1)
dataset
```

Out[20]:

	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perweek	Em
0	Urban	0	68	28561.99	Widowed	Male	0	7.978323	
1	Urban	1	27	21704.77	Married	Female	1	11.699080	
2	Urban	4	50	9609.57	Widowed	Female	0	10.752800	
3	Suburban	1	48	18925.23	Married	Male	0	14.913540	
4	Suburban	0	83	40074.19	Separated	Male	1	8.147417	
...	...	...	...	...	...	...	...	...	
9995	Rural	3	23	55723.74	Married	Male	0	9.415935	
9996	Rural	4	48	34129.34	Divorced	Male	0	6.740547	
9997	Rural	1	48	45983.43	Never Married	Female	0	6.590911	
9998	Urban	1	39	16667.58	Separated	Male	0	12.071910	
9999	Urban	1	28	9020.92	Never Married	Male	0	11.754720	

10000 rows × 55 columns

In [21]:
```
#Combining the dummy variables into the dataset
dataset = pd.concat([dataset, Contract_cat], axis = 1)
dataset
```

Out[21]:

	Area	Children	Age	Income	Marital	Gender	Churn	Outage_sec_perweek	Em
0	Urban	0	68	28561.99	Widowed	Male	0	7.978323	
1	Urban	1	27	21704.77	Married	Female	1	11.699080	
2	Urban	4	50	9609.57	Widowed	Female	0	10.752800	
3	Suburban	1	48	18925.23	Married	Male	0	14.913540	
4	Suburban	0	83	40074.19	Separated	Male	1	8.147417	
...	...	...	...	...	...	...	...	...	
9995	Rural	3	23	55723.74	Married	Male	0	9.415935	
9996	Rural	4	48	34129.34	Divorced	Male	0	6.740547	
9997	Rural	1	48	45983.43	Never Married	Female	0	6.590911	
9998	Urban	1	39	16667.58	Separated	Male	0	12.071910	
9999	Urban	1	28	9020.92	Never Married	Male	0	11.754720	

10000 rows × 58 columns

In [22]:
```
#drop area, gender, marital
dataset.drop(['Area', 'Gender', 'Marital', 'Contract','PaymentMethod',
dataset
```

Out[22]:

	Children	Age	Income	Churn	Outage_sec_perweek	Email	Contacts	Yearly_equip_fail
0	0	68	28561.99	0	7.978323	10	0	
1	1	27	21704.77	1	11.699080	12	0	
2	4	50	9609.57	0	10.752800	9	0	
3	1	48	18925.23	0	14.913540	15	2	
4	0	83	40074.19	1	8.147417	16	2	
...	...	...	...	...	...	...	...	
9995	3	23	55723.74	0	9.415935	12	2	
9996	4	48	34129.34	0	6.740547	15	2	
9997	1	48	45983.43	0	6.590911	10	0	
9998	1	39	16667.58	0	12.071910	14	1	
9999	1	28	9020.92	0	11.754720	17	1	

10000 rows × 48 columns

In [23]:
```
dataset.shape
```

Out[23]: (10000, 48)

In [24]:
```
#Export dataset to save as a checkpoint.
dataset.to_csv(r'churnClean2.csv', index = False)
```

In [25]:
```
y=dataset['Tenure'].values
```

In [26]:
```
y.shape
```

Out[26]: (10000,)

In [27]:
```
X=dataset.drop(columns=['Tenure'])
```

In [28]:
```
X.shape
```

Out[28]: (10000, 47)

Part IV: Analysis

The dataset was then split into train and test sets to prepare for the Random Forest model. The random forest, rf, was instantiated (cell 31). Next, a dictionary, params, was defined and a grid search was instantiated, fit to the training set and best hyperparameters were selected from the Grid Search. The data was fit to the training set and prediction run on the 'X_test' data. Next, an RMSE was found, resulting in a score of 18.05. This RMSE is much too high and is over 24% of the range of the dependent variable Tenure. (Tenure's range is approximately 71). Root Mean Squared Error (RMSE) and Mean Square Error (MSE) measures "how well the model fits dependent variables" (Wu, 2020).

RMSE is simply the square root of MSE, and "brings it back to the same level of prediction error and makes it easier for interpretation…It gives you an absolute number on how much your predicted results deviate from the actual result" (Wu, 2020). As previously mentioned, the RMSE on this model is 18.05, which is over 24% of the dependent variable Tenure's range. Making predictions that vary up to 24% of the dependent variable range does not have accurate predictive value and are the result of an inaccurate model.

In [29]:
```
SEED=1
```

In [30]:
```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.3,
```

In [31]:
```
rf=RandomForestRegressor(n_estimators=400, min_samples_leaf=0.12, ranc
```

In [32]:
```
params_rf={
        'n_estimators':[300,400,500],
        'max_depth':[4,6,8],
        'min_samples_leaf':[0.1, 0.2],
        'max_features':['log2', 'sqrt']
}
```

In [33]:
```
#instantiate 'grid_rf'
grid_rf=GridSearchCV(estimator=rf,
                     param_grid=params_rf,
                     cv=3,
                     scoring='neg_mean_squared_error',
                     verbose=1,
                     n_jobs=-1)
```

In [34]:
```
#fit grid_rf to training set
grid_rf.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 36 candidates, totalling 108 fits
```

Out[34]:
```
GridSearchCV(cv=3,
             estimator=RandomForestRegressor(min_samples_leaf=0.12,
                                             n_estimators=400, random
_state=1),
             n_jobs=-1,
             param_grid={'max_depth': [4, 6, 8],
                         'max_features': ['log2', 'sqrt'],
                         'min_samples_leaf': [0.1, 0.2],
                         'n_estimators': [300, 400, 500]},
             scoring='neg_mean_squared_error', verbose=1)
```

In [35]:
```
#extract the best hyperparamters from grid_rf
best_hyperparams=grid_rf.best_params_
```

In [36]:
```
print('Best hyperparameters :\n', best_hyperparams)
```

```
Best hyperparameters :
 {'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 0.1, 'n
_estimators': 300}
```

In [37]:
```
#extract best model from 'grid_rf'
best_model=grid_rf.best_estimator_
```

In [38]:
```
y_pred=best_model.predict(X_test)
```

In [39]:
```
rmse_test=MSE(y_test, y_pred)**(1/2)
```

In [40]:
```
print('Test set RMSE of rf:{:.2f}'.format(rmse_test))
```

```
Test set RMSE of rf:18.05
```

Part V: Data Summary and Implications

In summary, the independent variables used in this Random Forest model do not yield predictive results for Tenure. The RMSE is much too high. The implications of this analysis revolve around the fact that this organization cannot predict whether a customer tenure length using the independent variables described in this analysis. One limitation of this analysis is the potential presence of other variables, not in the dataset, that can affect customer tenure. This organization should continue to investigate the independent variables, or combinations thereof, that could be predictive of tenure.

References

Lorberfeld, A. (2019, March 26). Machine Learning Algorithms In Layman's Terms, Part 2. Retrieved November 14, 2021, from https://towardsdatascience.com/machine-learning-algorithms-in-laymans-terms-part-2-a0a74df9a9ac

Wu, S. (2020, June 14). *What are the best metrics to evaluate your regression model?* Medium. https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b