Tree-based ML and Feature Selection

This week, your assignment is:

- use our prepared churn data from week 2
- break our data into features and targets, and train and test sets
- use sklearn to fit a decision tree to the training data
  - plot the decision tree
  - change the max_depth of the decision tree to improve the model if needed (or tune it with a hyperparameter search)
- plot the correlations between features and targets
- use sklearn to fit a random forest model to predict churn from our dataset
  - plot the feature importances from the random forest
- choose some of the less-important features to remove from the model using feature importances and correlations and fit the random forest model to the new data
  - examine the feature importances after removing less important features
- write a short analysis of the results of your work

***Optional* advanced tasks:**

- use H2O to fit a random forest to our original, unmodified data (missing values and all)
  - you can decide if you want to break the data into train and test sets or not, but remember it's best to evaluate performance on a test or validation dataset
  - plot the H2O random forest's feature importances
- tune the random forest hyperparameters for the sklearn and/or H2O models
- use forward and/or backward selection with feature importances from a random forest model
- use recursive feature selection
- compare the various feature selection methods you tried and write a short summary

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn import tree
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv("/Users/joshaiken/Desktop/MSDS_Homework/MSDS600_X40/Week_2/prepped_c
df = df.drop(columns = ["Unnamed: 0", "customerID"])
df
```

Out[3]:

	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Churn	total_charges_tenu
0	1	0	0	1	29.85	29.85	0	0
1	34	1	1	0	56.95	1889.50	0	0
2	2	1	0	0	53.85	108.15	1	0
3	45	0	1	3	42.30	1840.75	0	0
4	2	1	0	1	70.70	151.65	1	0
...	...	...	...	...	...	...	...	
7038	24	1	1	0	84.80	1990.50	0	0
7039	72	1	1	2	103.20	7362.90	0	0
7040	11	0	0	1	29.60	346.45	0	0
7041	4	1	0	0	74.40	306.60	1	0
7042	66	1	2	3	105.65	6844.50	0	0

7043 rows × 8 columns

In [4]:
```
features = df.drop("Churn", axis = 1)
targets = df["Churn"]

x_train, x_test, y_train, y_test = train_test_split(features, targets, stratify = tar
```

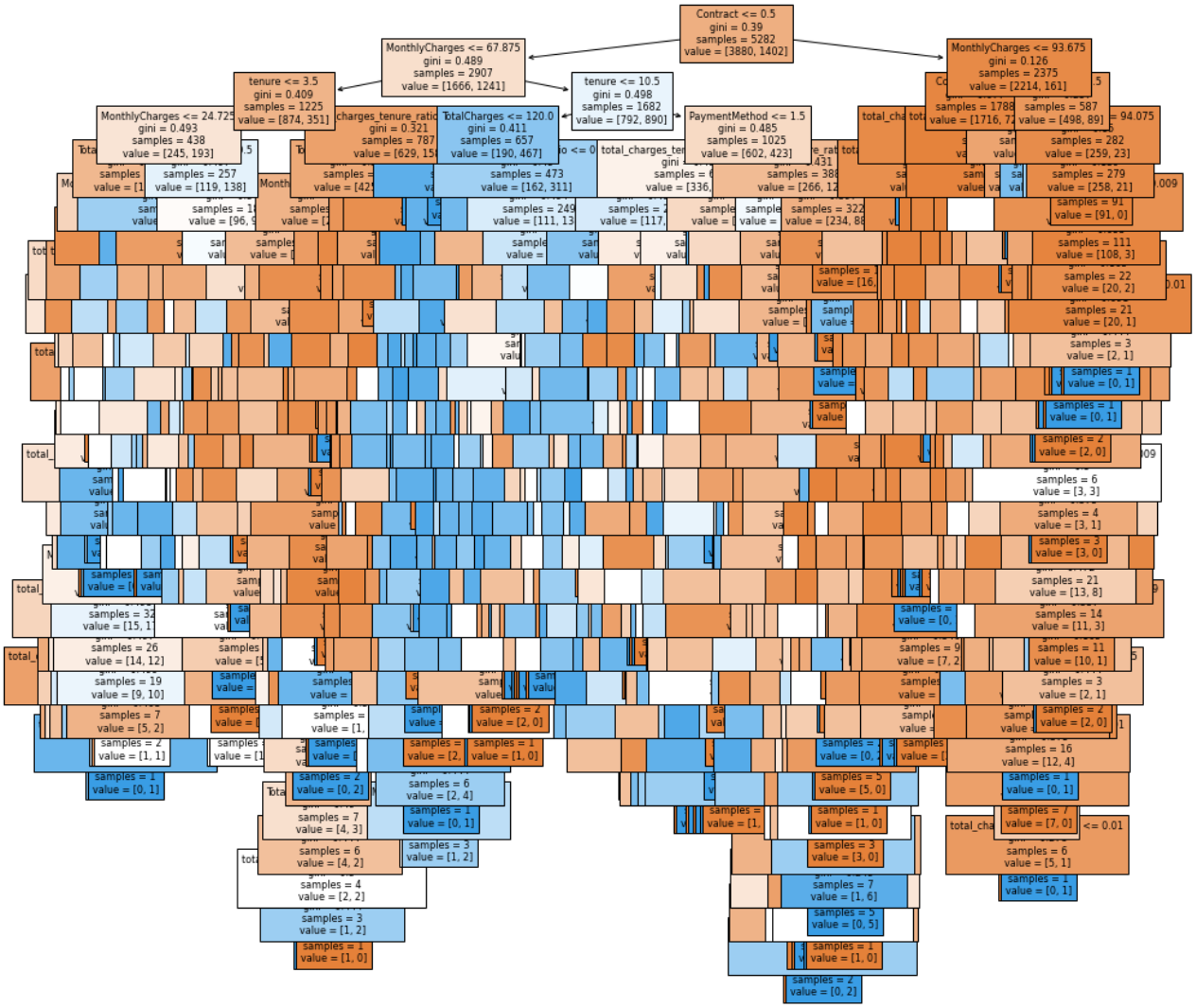
In [5]:
```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

print(dt.score(x_train, y_train))
print(dt.score(x_test, y_test))
```

```
0.993563044301401
0.7268597387847814
```

In [6]:
```
dt.get_depth()
```

Out[6]:
```
28
```

In [7]:
```
f = plt.figure(figsize=(15, 15))
_ = tree.plot_tree(dt, fontsize=8, feature_names=features.columns, filled=True)
```



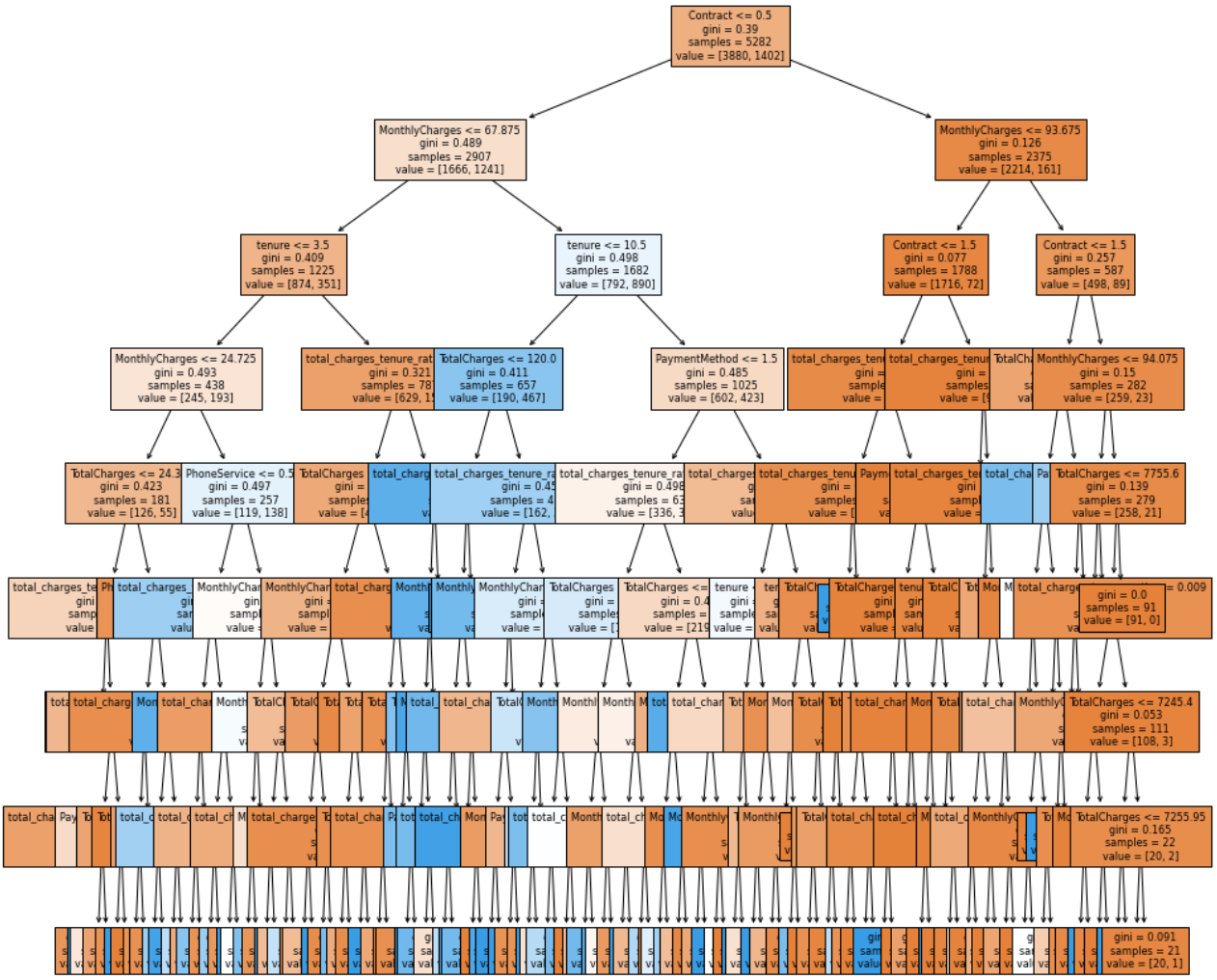
This is definitely not what I had expected, originally... But as I thought about it, I had changed a lot of the columns to simply 1 or 0, so I am speculating that this was the reason for having a depth of 28...

In [8]:
```
dt = DecisionTreeClassifier(max_depth= 8)
dt.fit(x_train, y_train)
print(dt.score(x_train, y_train))
print(dt.score(x_test, y_test))
```

```
0.8296099962135555
0.7739920499716071
```

A depth of 8 seemed to give the best result on the test set while giving the best possible result on the train set given the goal of a higher test set result

In [9]:
```
f = plt.figure(figsize=(15, 15))
_ = tree.plot_tree(dt, fontsize=8, feature_names=features.columns, filled=True)
```



A depth of 8 still gives a very large overall tree.

In [10]:
```
X, y = df, targets
crossvalidation = KFold(n_splits=6, shuffle=True, random_state=1)
for depth in range(1,8):
    tree_classifier = tree.DecisionTreeClassifier(max_depth = depth, random_state = 0
    if tree_classifier.fit(X,y).tree_.max_depth < depth:
        break
    score = np.mean(cross_val_score(tree_classifier, X, y, scoring = "accuracy", cv =
    print("Depth: %i Accuracy: %.3f" % (depth, score))
```

```
Depth: 1 Accuracy: 1.000
```

In [11]:
```
from sklearn.ensemble import RandomForestClassifier
import math
rfc = RandomForestClassifier(max_depth=8, random_state=42)
rfc.fit(x_train, y_train)

print(rfc.score(x_train, y_train))
print(rfc.score(x_test, y_test))
```

```
0.8366149185914427
0.7932992617830777
```

In [12]:
```
math.sqrt(x_train.shape[1])
```

Out[12]:
```
2.6457513110645907
```

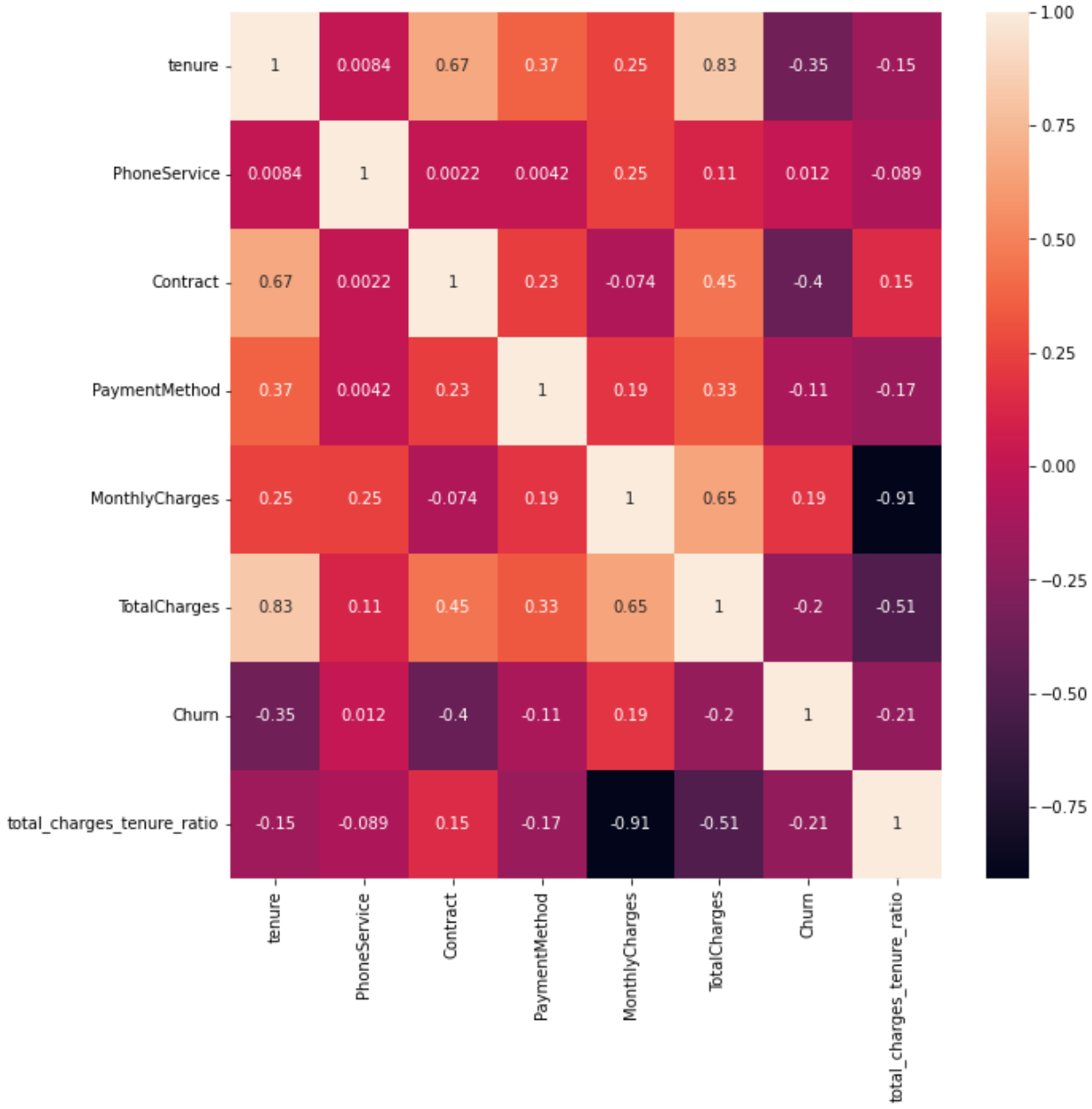
In [13]:
```
rfc = RandomForestClassifier(max_depth=8, max_features=3, random_state=42)
rfc.fit(x_train, y_train)

print(rfc.score(x_train, y_train))
print(rfc.score(x_test, y_test))
```

```
0.8407800075728891
0.7927314026121521
```

In [14]:
```
import seaborn as sns
```

In [15]:
```
f = plt.figure(figsize=(10, 10))
sns.heatmap(df.corr(), annot=True)
```

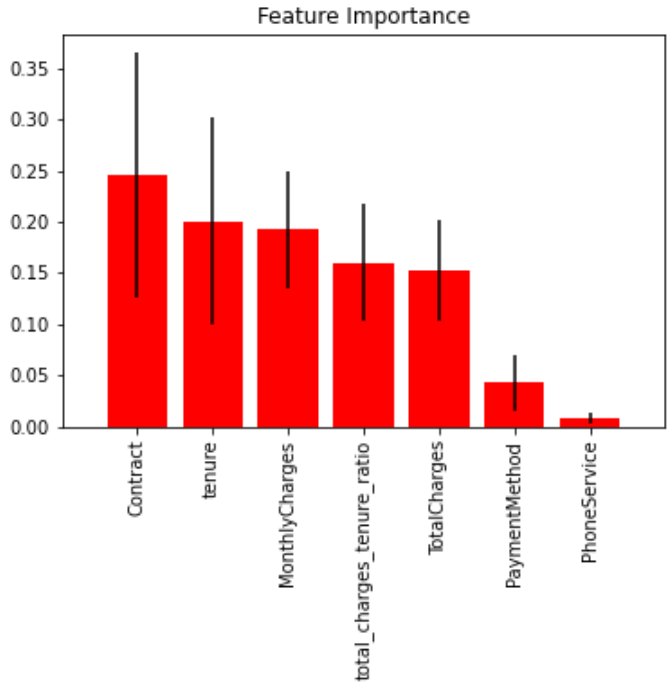
Out[15]: <AxesSubplot:>



This heatmap shows a few interesting things. Tenure, contract, monthly charges, and total charges all have an impact on churn. Customers that spend more are more invested into the relationship with their vendor. The same can be send about customers who are willing to enter into a contract or prefer to go monthly. These all dictate how long a customer stays and the longer a customer stays, the more likely they are to continue staying with a vendor.

In [20]:
```
from scikitplot.estimators import plot_feature_importances

plot_feature_importances(rfc, feature_names=features.columns, x_tick_rotation=90)
```

Out[20]:
```
<AxesSubplot:title={'center':'Feature Importance'}>
```



This shows the obvious, and helps to validate the heatmap. Whether or not a customer is in contract, and how longer they have been with the vendor are the biggest indicators of whether a customer will churn or not.

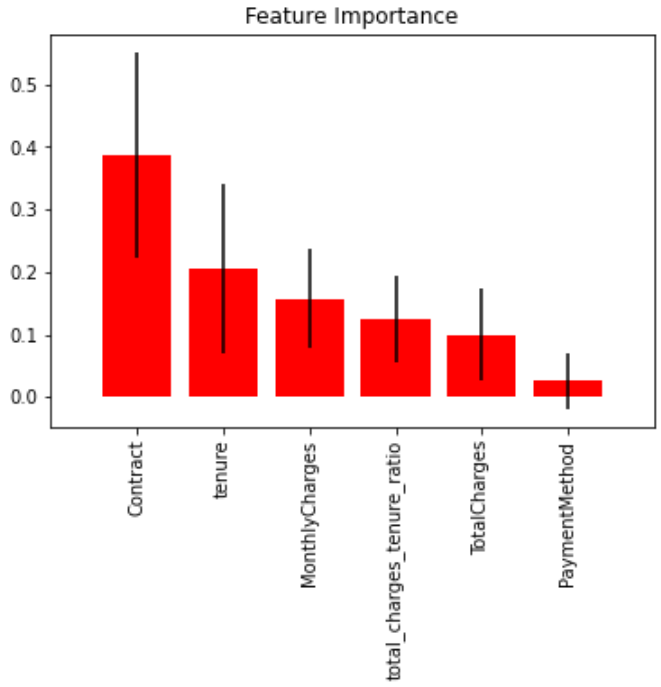
In [21]:
```
new_features = features.drop(["PhoneService"], axis = 1)

x_train, x_test, y_train, y_test = train_test_split(new_features, targets, stratify=t
```

In [31]:
```
rfc = RandomForestClassifier(max_depth=5, max_features=3, random_state=42)
rfc.fit(x_train, y_train)

print(rfc.score(x_train, y_train))
print(rfc.score(x_test, y_test))
```

```
0.8080272624006059
0.7910278250993753
```

In [32]:
```
plot_feature_importances(rfc, feature_names=new_features.columns, x_tick_rotation=90)
```

Out[32]:
```
<AxesSubplot:title={'center':'Feature Importance'}>
```



Dropping Phone Service as a feature was able to help further bridge the gap between the train and test sets. Not by much, but we still did make a little progress. The thing that it did change was the feature importance. It further separated Contract as the primary feature and increased separation between tenure and monthly charges.

```
This process started a little overwhelming, with our first decision tree going down
to 28 branches. We had to do a little further adjustment with the model and
ultimately removing the phone service as a feature as the impact on that feature was
far lower than I expected. What these models show us is that the biggest factors in
relation to churn are Contract, Tenure, and how much a company spends, monthly, and
over the life of their tenure. This all makes sense. A contract keeps a customer
locked in with the company, regardless of spend or tenure. But customers that are in
contract, and continue to renew their contracts are the best situation for a vendor.
If a customer is month-to-month, and has not been with the vendor for long, they are
at a higher risk to churn.
```

Summary

Be sure to write a summary of your work and explain the results.