In [1]:
```
# Import libraries
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error

# Import dataset
df = pd.read_csv("C:/Users/hkeim/OneDrive/Documents/School/D209/churn_clean.csv")
```

In [2]:
```
# Drop extraneous variables
df = df.drop(['Customer_id', 'Interaction', 'UID', 'City', 'County', 'Job'], axis=1)
```

In [3]:
```
# Change object type to category codes
df['State'] = df['State'].astype('category')
df['State'] = df['State'].cat.codes
```

In [4]:
```
df['Area'] = df['Area'].astype('category')
df['Area'] = df['Area'].cat.codes
```

In [5]:
```
df['TimeZone'] = df['TimeZone'].astype('category')
df['TimeZone'] = df['TimeZone'].cat.codes
```

In [6]:
```
df['Marital'] = df['Marital'].astype('category')
df['Marital'] = df['Marital'].cat.codes
```

In [7]:
```
df['Gender'] = df['Gender'].astype('category')
df['Gender'] = df['Gender'].cat.codes
```

In [8]:
```
df['Contract'] = df['Contract'].astype('category')
df['Contract'] = df['Contract'].cat.codes
```

In [9]:
```
df['PaymentMethod'] = df['PaymentMethod'].astype('category')
df['PaymentMethod'] = df['PaymentMethod'].cat.codes
```

In [10]:
```
df['InternetService'] = df['InternetService'].astype('category')
df['InternetService'] = df['InternetService'].cat.codes
```

In [11]:
```
# Create dummies for bianry objects
df = pd.get_dummies(df, columns = ['Churn', 'Techie', 'Port_modem', 'Tablet', 'Phone', 'Multi
                                   'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSup
# Drop 'No' dummies
df = df.drop(['Churn_No', 'Techie_No', 'Port_modem_No', 'Tablet_No', 'Phone_No', 'Multiple_No
              'StreamingTV_No', 'StreamingMovies_No', 'PaperlessBilling_No'], axis=1)
```

In [12]:
```
# Set seed
```

```
SEED = 1
```

In [13]:
```
# Create features array
X = df.drop('Tenure', axis = 1)
```

In [14]:
```
# Create target array
y = df['Tenure']
```

In [15]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2, random_state = SEED
```

In [16]:
```
# Instantiate rf
rf = RandomForestRegressor()

# Fit rf to the training set
rf.fit(X_train, y_train)
```

Out[16]:
```
RandomForestRegressor()
```

In [17]:
```
# Define the dictionary 'params_rf'
params_rf = {'n_estimators':[100, 350, 500],
             'max_features':['log2', 'auto', 'sqrt'],
             'min_samples_leaf':[2, 10, 30]}
```

In [18]:
```
# Instantiate grid_rf
grid_rf = GridSearchCV(estimator = rf,
                       param_grid = params_rf,
                       scoring ='neg_mean_squared_error',
                       cv = 3,
                       verbose = 1,
                       n_jobs = -1)
```

In [19]:
```
# Fit grid_rf
grid_rf.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
```

Out[19]:
```
GridSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'max_features': ['log2', 'auto', 'sqrt'],
                         'min_samples_leaf': [2, 10, 30],
                         'n_estimators': [100, 350, 500]},
             scoring='neg_mean_squared_error', verbose=1)
```

In [20]:
```
# Extract the best estimator
best_model = grid_rf.best_estimator_

# Predict y values
y_pred = best_model.predict(X_test)

# Print best_model params
print(best_model.get_params)
```

```
<bound method BaseEstimator.get_params of RandomForestRegressor(min_samples_leaf=2, n_estimato
rs=350)>
```

In [21]:
```
# Evaluate best model
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', (metrics.mean_squared_error(y_test, y_pred))**(1/2))
print('R2 Score:', metrics.r2_score(y_test, y_pred))
```

```
MSE: 1.5005805573052682
RMSE: 1.224981859990289
R2 Score: 0.9978687165707069
```

In [22]:
```
# Save CSV file of prepared data
df.to_csv("C:/Users/hkeim/OneDrive/Documents/School/D209/Keim D209 Task Two Clean Data.csv",
```

In [23]:
```
#create unified test and tran sets for export
trainframes = [X_train, y_train]
train = pd.concat(trainframes)
testframes = [X_test, y_test]
test = pd.concat(testframes)
```

In [24]:
```
# Save CSV files of training and testing data
train.to_csv("C:/Users/hkeim/OneDrive/Documents/School/D209/Keim D209 Task Two Training Data.
test.to_csv("C:/Users/hkeim/OneDrive/Documents/School/D209/Keim D209 Task Two Testing Data.cs
```

In [ ]: