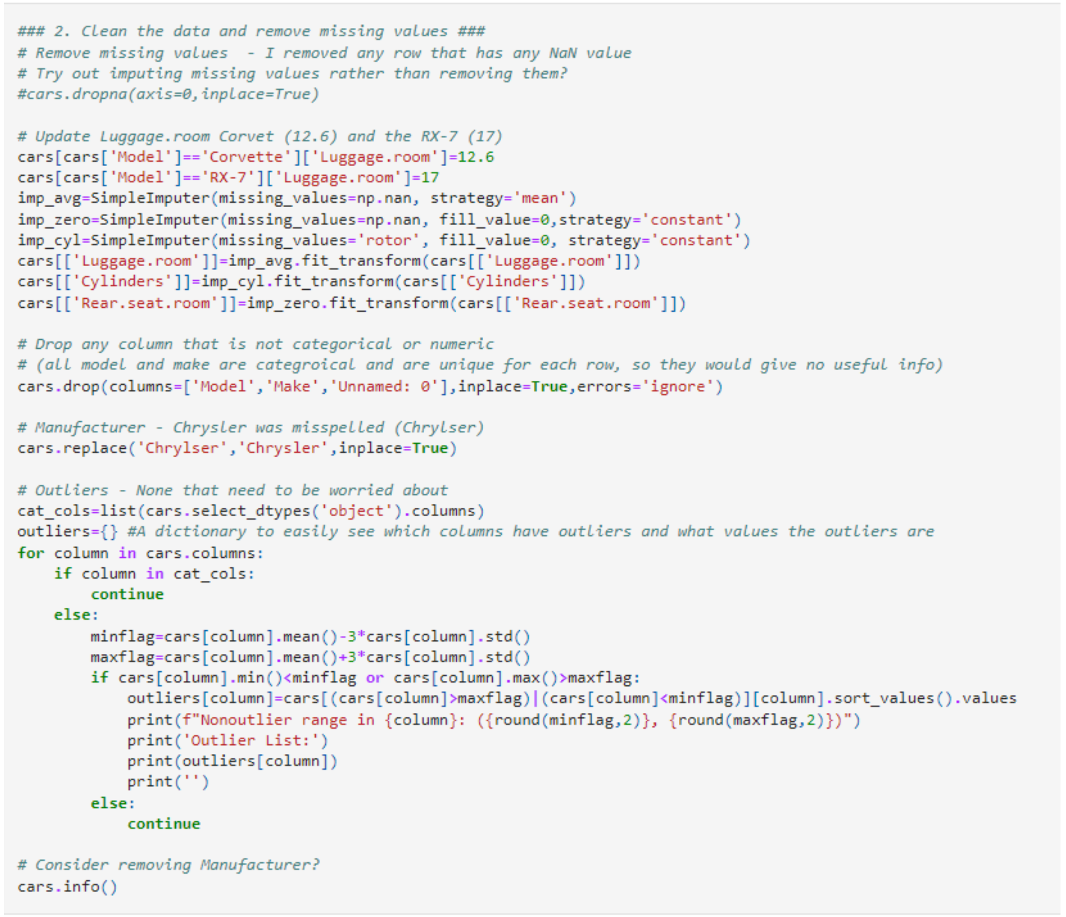
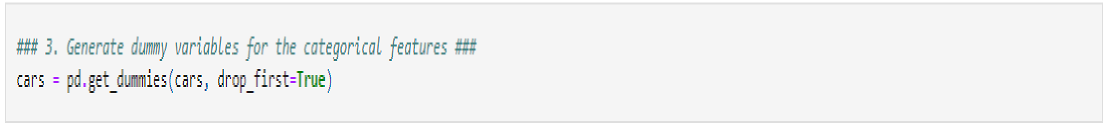
Run (random) Forest Run

Sargis Abrahamyan, Justin Bartell,
Raymond Sepulveda, Daniel Brickman

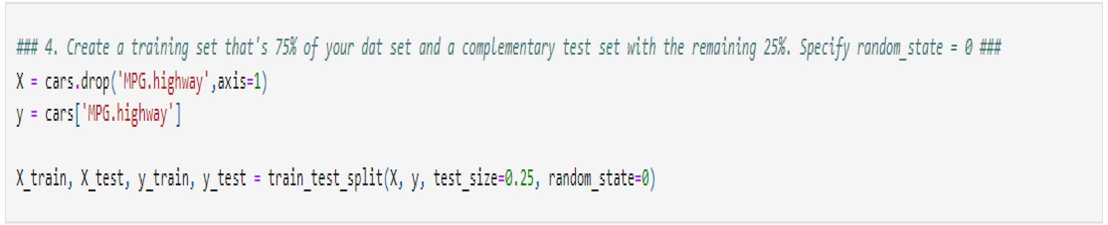
What's Normal Anyways?



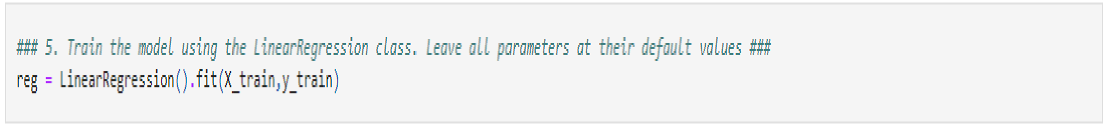
```
### 2. Clean the data and remove missing values ###
# Remove missing values  - I removed any row that has any NaN value
# Try out imputing missing values rather than removing them?
#cars.dropna(axis=0,inplace=True)

# Update Luggage.room Corvet (12.6) and the RX-7 (17)
cars[cars['Model']=='Corvette']['Luggage.room']=12.6
cars[cars['Model']=='RX-7']['Luggage.room']=17
imp_avg=SimpleImputer(missing_values=np.nan, strategy='mean')
imp_zero=SimpleImputer(missing_values=np.nan, fill_value=0,strategy='constant')
imp_cyl=SimpleImputer(missing_values='rotor', fill_value=0, strategy='constant')
cars[['Luggage.room']]=imp_avg.fit_transform(cars[['Luggage.room']])
cars[['Cylinders']]=imp_cyl.fit_transform(cars[['Cylinders']])
cars[['Rear.seat.room']]=imp_zero.fit_transform(cars[['Rear.seat.room']])

# Drop any column that is not categorical or numeric
# (all model and make are categroical and are unique for each row, so they would give no useful info)
cars.drop(columns=['Model','Make','Unnamed: 0'],inplace=True,errors='ignore')

# Manufacturer - Chrysler was misspelled (Chrylser)
cars.replace('Chrylser','Chrysler',inplace=True)

# Outliers - None that need to be worried about
cat_cols=list(cars.select_dtypes('object').columns)
outliers={} #A dictionary to easily see which columns have outliers and what values the outliers are
for column in cars.columns:
    if column in cat_cols:
        continue
    else:
        minflag=cars[column].mean()-3*cars[column].std()
        maxflag=cars[column].mean()+3*cars[column].std()
        if cars[column].min()<minflag or cars[column].max()>maxflag:
            outliers[column]=cars[(cars[column]>maxflag)|(cars[column]<minflag)][column].sort_values().values
            print(f"Nonoutlier range in {column}: ({round(minflag,2)}, {round(maxflag,2)})")
            print('Outlier List:')
            print(outliers[column])
            print('')
        else:
            continue

# Consider removing Manufacturer?
cars.info()
```



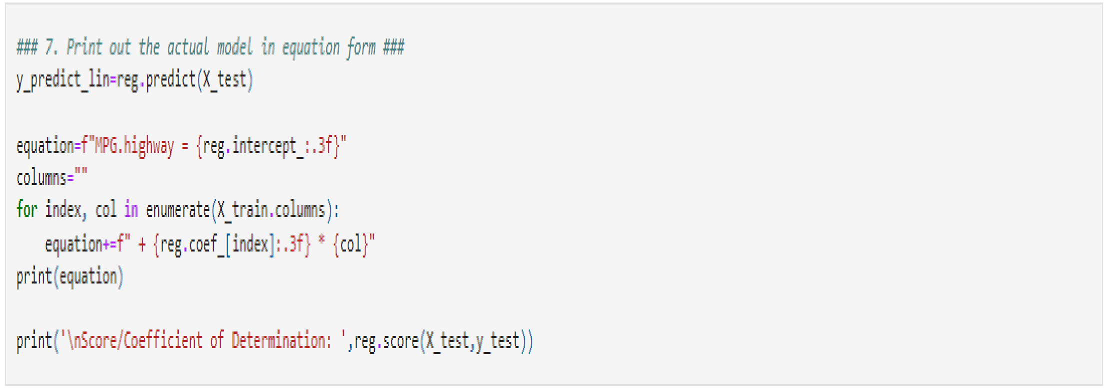
```
### 3. Generate dummy variables for the categorical features ###
cars = pd.get_dummies(cars, drop_first=True)
```



```
### 4. Create a training set that's 75% of your dat set and a complementary test set with the remaining 25%. Specify random_state = 0 ###
X = cars.drop('MPG.highway',axis=1)
y = cars['MPG.highway']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

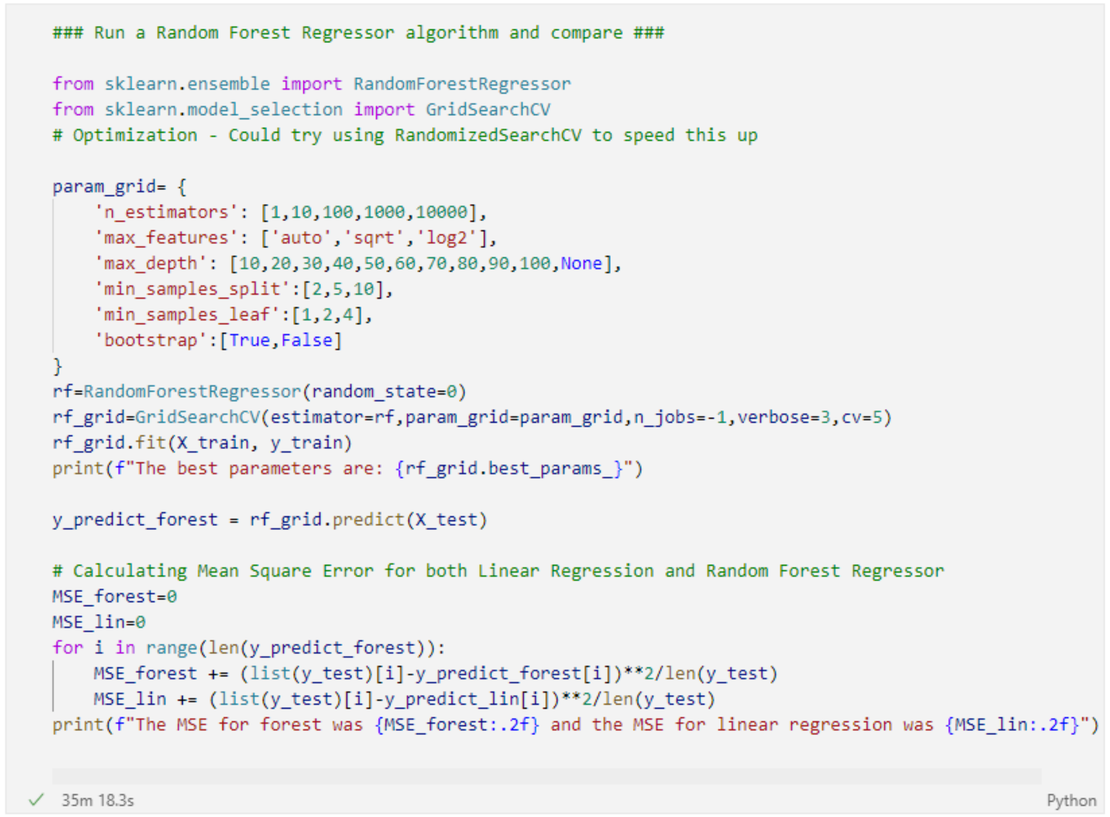


```
### 5. Train the model using the LinearRegression class. Leave all parameters at their default values ###
reg = LinearRegression().fit(X_train,y_train)
```



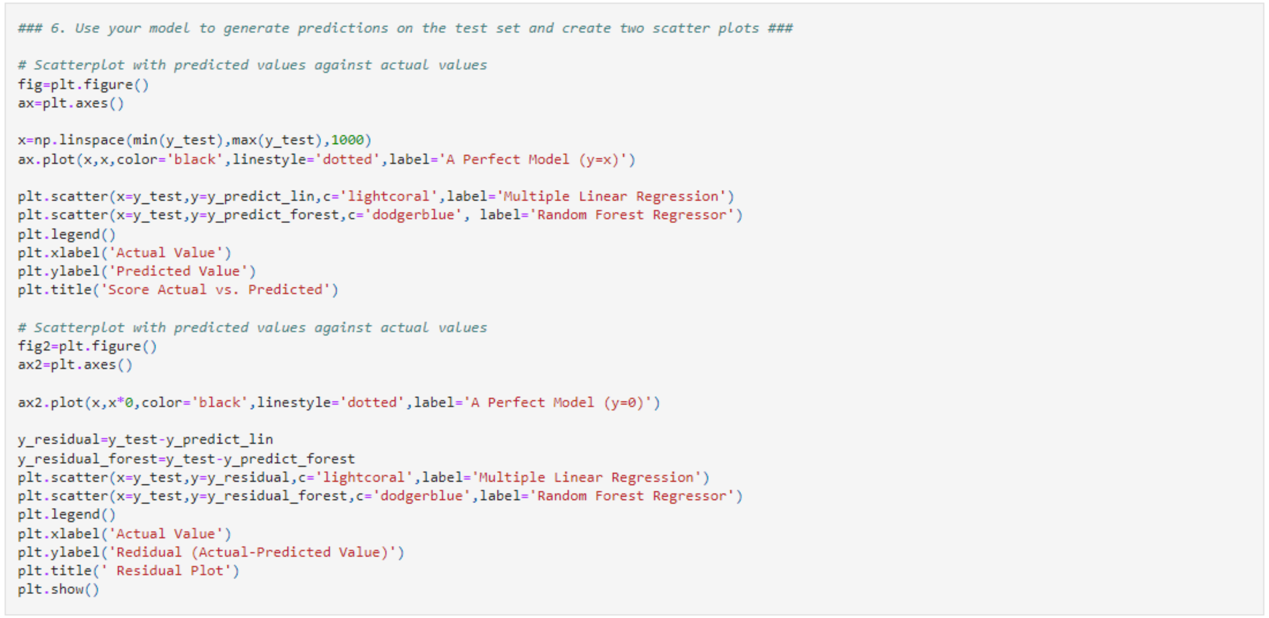
```
### 7. Print out the actual model in equation form ###
y_predict_lin=reg.predict(X_test)

equation=f"MPG.highway = {reg.intercept_:.3f}"
columns=""
for index, col in enumerate(X_train.columns):
    equation+=f" + {reg.coef_[index]:.3f} * {col}"
print(equation)

print('\nScore/Coefficient of Determination: ',reg.score(X_test,y_test))
```

This had run for 3 years, 2 months, 14 days, 16 hours, **35 minutes and 18.3 seconds**



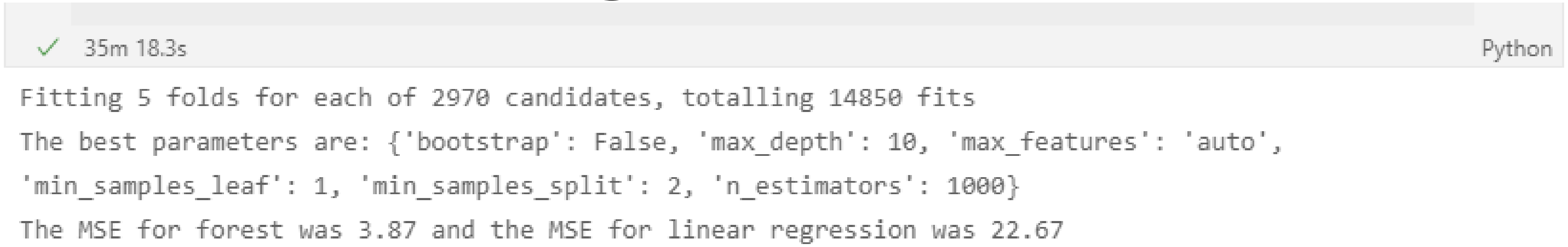
```python
### Run a Random Forest Regressor algorithm and compare ###

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
# Optimization - Could try using RandomizedSearchCV to speed this up

param_grid= {
    'n_estimators': [1,10,100,1000,10000],
    'max_features': ['auto','sqrt','log2'],
    'max_depth': [10,20,30,40,50,60,70,80,90,100,None],
    'min_samples_split':[2,5,10],
    'min_samples_leaf':[1,2,4],
    'bootstrap':[True,False]
}
rf=RandomForestRegressor(random_state=0)
rf_grid=GridSearchCV(estimator=rf,param_grid=param_grid,n_jobs=-1,verbose=3,cv=5)
rf_grid.fit(X_train, y_train)
print(f"The best parameters are: {rf_grid.best_params_}")

y_predict_forest = rf_grid.predict(X_test)

# Calculating Mean Square Error for both Linear Regression and Random Forest Regressor
MSE_forest=0
MSE_lin=0
for i in range(len(y_predict_forest)):
    MSE_forest += (list(y_test)[i]-y_predict_forest[i])**2/len(y_test)
    MSE_lin += (list(y_test)[i]-y_predict_lin[i])**2/len(y_test)
print(f"The MSE for forest was {MSE_forest:.2f} and the MSE for linear regression was {MSE_lin:.2f}")
```

✓ 35m 18.3s Python



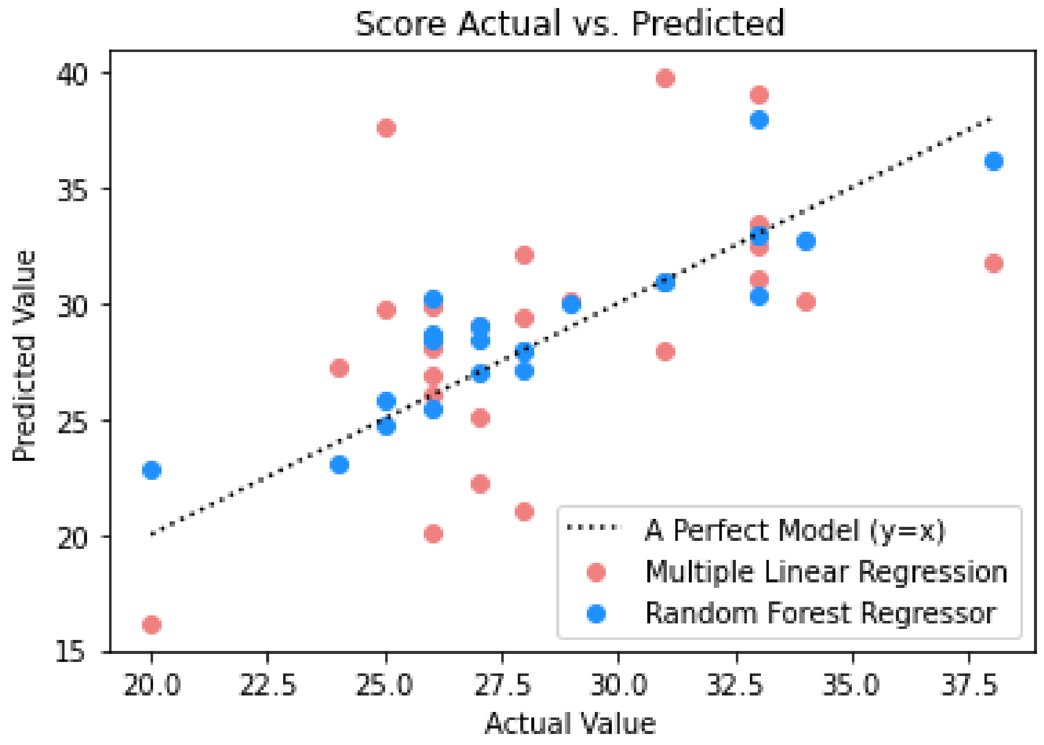
```python
### 6. Use your model to generate predictions on the test set and create two scatter plots ###

# Scatterplot with predicted values against actual values
fig=plt.figure()
ax=plt.axes()

x=np.linspace(min(y_test),max(y_test),1000)
ax.plot(x,x,color='black',linestyle='dotted',label='A Perfect Model (y=x)')

plt.scatter(x=y_test,y=y_predict_lin,c='lightcoral',label='Multiple Linear Regression')
plt.scatter(x=y_test,y=y_predict_forest,c='dodgerblue', label='Random Forest Regressor')
plt.legend()
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.title('Score Actual vs. Predicted')

# Scatterplot with predicted values against actual values
fig2=plt.figure()
ax2=plt.axes()

ax2.plot(x,x*0,color='black',linestyle='dotted',label='A Perfect Model (y=0)')

y_residual=y_test-y_predict_lin
y_residual_forest=y_test-y_predict_forest
plt.scatter(x=y_test,y=y_residual,c='lightcoral',label='Multiple Linear Regression')
plt.scatter(x=y_test,y=y_residual_forest,c='dodgerblue',label='Random Forest Regressor')
plt.legend()
plt.xlabel('Actual Value')
plt.ylabel('Redidual (Actual-Predicted Value)')
plt.title(' Residual Plot')
plt.show()
```

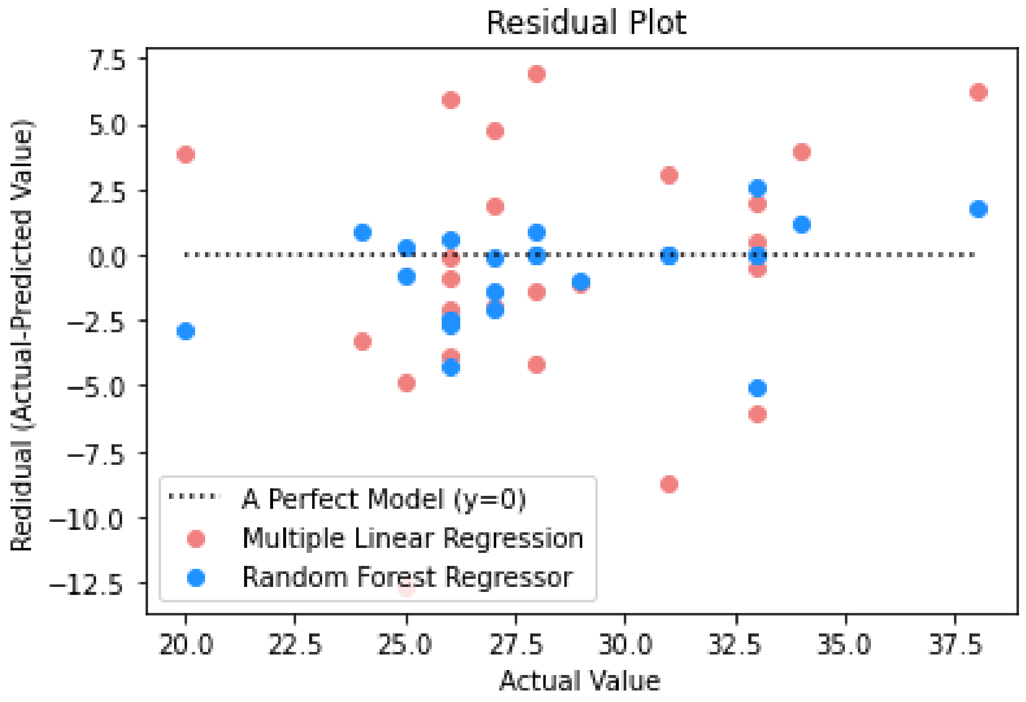
Forrest and Jenny Goes Together like Multiple Linear Regression and Random Forest



✓ 35m 18.3s Python

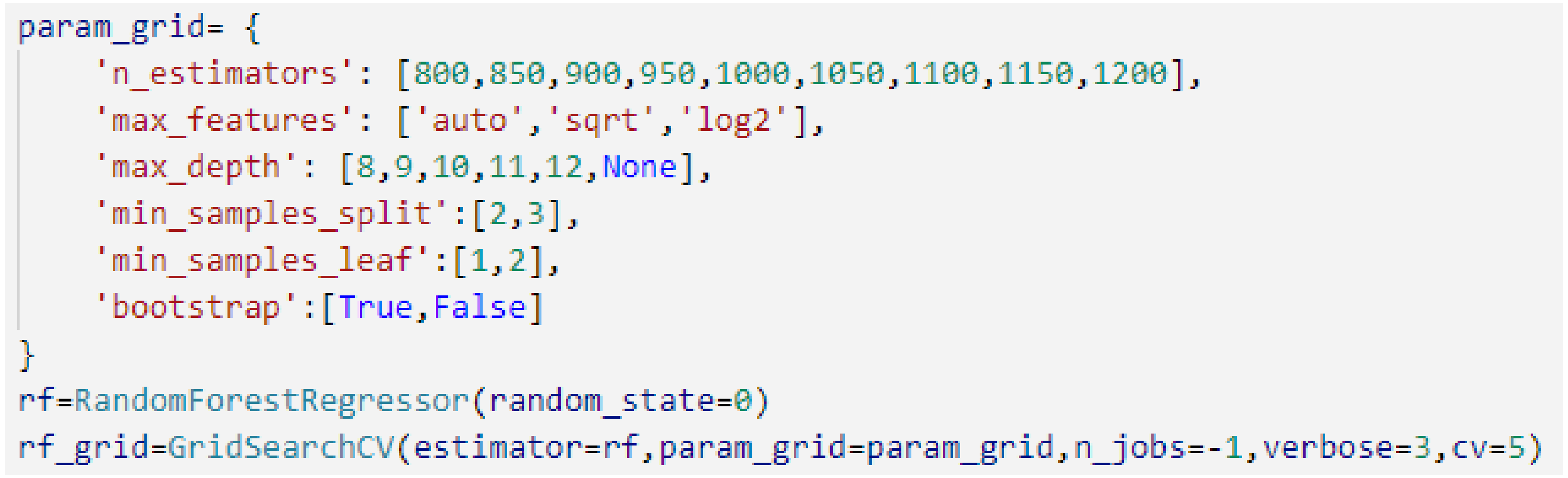
```
Fitting 5 folds for each of 2970 candidates, totalling 14850 fits
The best parameters are: {'bootstrap': False, 'max_depth': 10, 'max_features': 'auto',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 1000}
The MSE for forest was 3.87 and the MSE for linear regression was 22.67
```





I never thanked you for making my model better

- Get more data



```
param_grid= {
    'n_estimators': [800,850,900,950,1000,1050,1100,1150,1200],
    'max_features': ['auto','sqrt','log2'],
    'max_depth': [8,9,10,11,12,None],
    'min_samples_split':[2,3],
    'min_samples_leaf':[1,2],
    'bootstrap':[True,False]
}
rf=RandomForestRegressor(random_state=0)
rf_grid=GridSearchCV(estimator=rf,param_grid=param_grid,n_jobs=-1,verbose=3,cv=5)
```