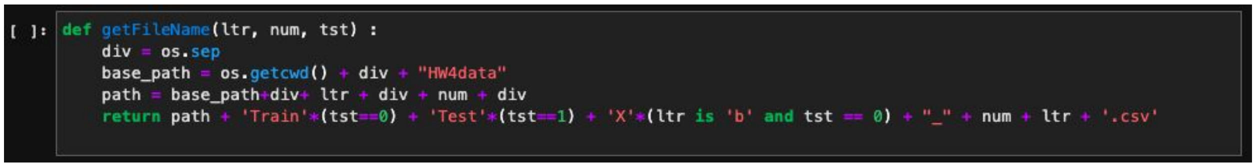
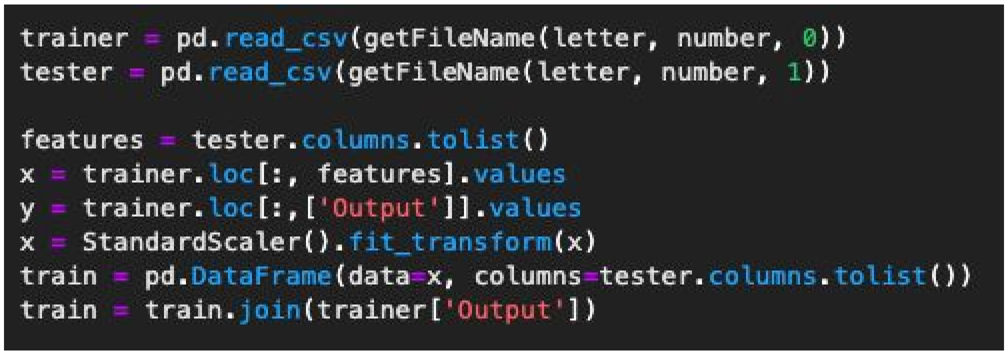
**Drew Gottlieb**
**CS 123 HW 4 Report**
**10-27-20**
In this report, I will walk through my code and my thought process while tackling this assignment.

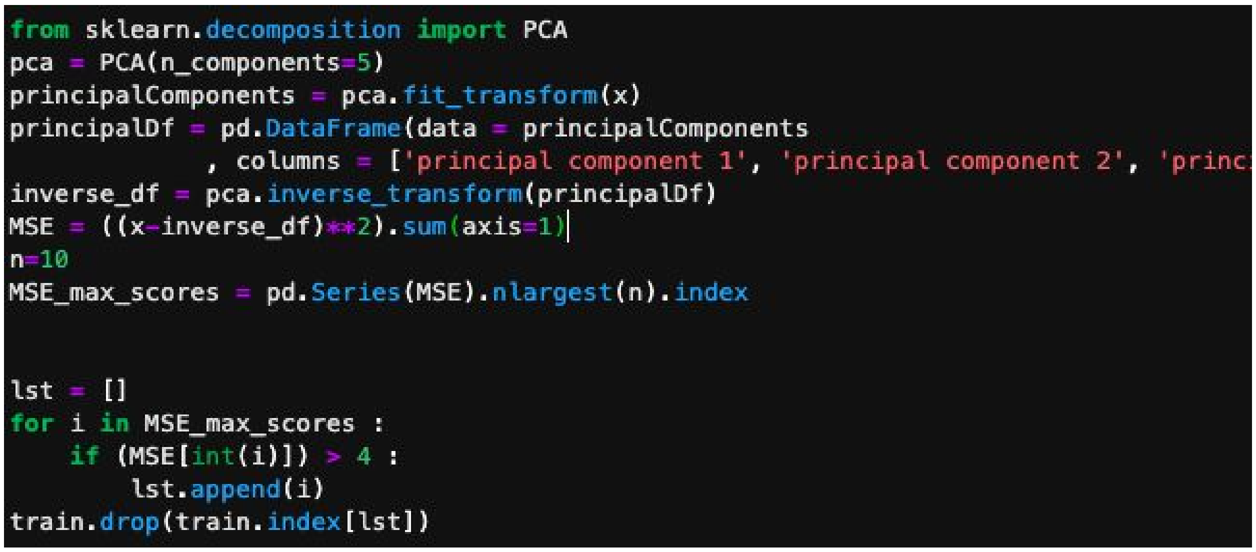
I tested a few libraries before deciding to simply use sklearn's LinearRegressor, as a linear regression was suggested by the professor. I was considering using Random Forest, but this algorithm is prone to overfitting, and I didn't have the time to fine-tune the parameters, and I'm not even sure there's enough data to verify that I wouldn't be overfitting. The sparsity of the data also stopped me from trying a more complex library like LGBM, as these do not really do well when the training sets are only like 200 data points.


```
[ ]: def getFileName(ltr, num, tst) :
         div = os.sep
         base_path = os.getcwd() + div + "HW4data"
         path = base_path+div+ ltr + div + num + div
         return path + 'Train'*(tst==0) + 'Test'*(tst==1) + 'X'*(ltr is 'b' and tst == 0) + "_" + num + ltr + '.csv'
```

This is a simple helper method that gets the names of the files based on the letter, the number, and tst which is a binary variable that gets whether the requested file is training or testing. This is important because the files in the b folder had X after some of the names.


```
trainer = pd.read_csv(getFileName(letter, number, 0))
tester = pd.read_csv(getFileName(letter, number, 1))

features = tester.columns.tolist()
x = trainer.loc[:, features].values
y = trainer.loc[:,['Output']].values
x = StandardScaler().fit_transform(x)
train = pd.DataFrame(data=x, columns=tester.columns.tolist())
train = train.join(trainer['Output'])
```
I started by scaling the data so it would have a mean of 0 and a standard deviation of 1. I did not scale the output column, as this would skew my predictions. Through this, I am simply trying to find out which columns are causing the most variation in the data.

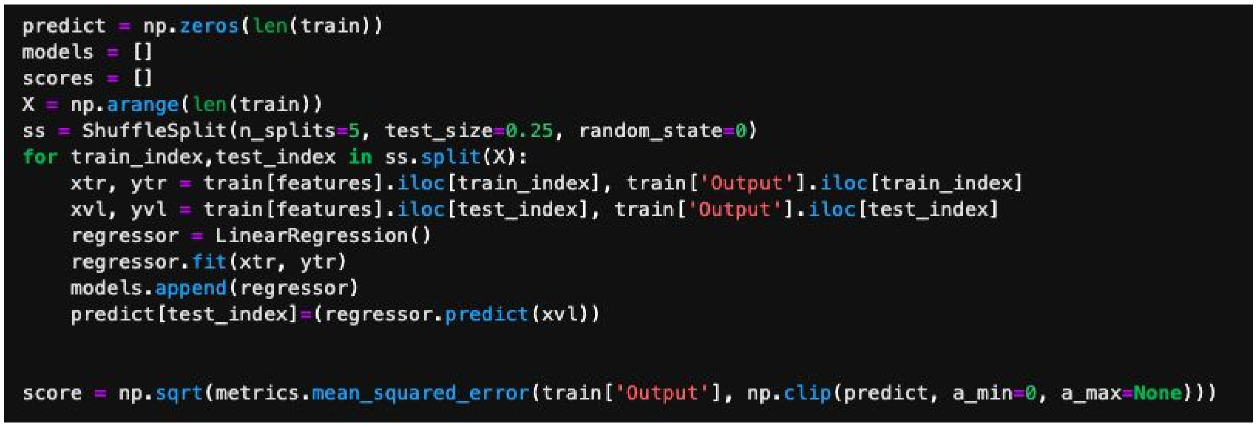


```
from sklearn.decomposition import PCA
pca = PCA(n_components=5)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2', 'princ
inverse_df = pca.inverse_transform(principalDf)
MSE = ((x-inverse_df)**2).sum(axis=1)
n=10
MSE_max_scores = pd.Series(MSE).nlargest(n).index


lst = []
for i in MSE_max_scores :
    if (MSE[int(i)]) > 4 :
        lst.append(i)
train.drop(train.index[lst])
```

Here, I use PCA decomposition in order to find out where the most variance is in the data. I also use this to eliminate outliers. After manually experimenting with the data, I found that 5 components captured around 85% of the variation in the data.

I use the inverse transform for the PCA and then take a simple RMSE to find out which points are the biggest outliers. I remove any point with an RMSE over 4. This was kind of arbitrary, and I wish I had more time to experiment with the exact parameters.

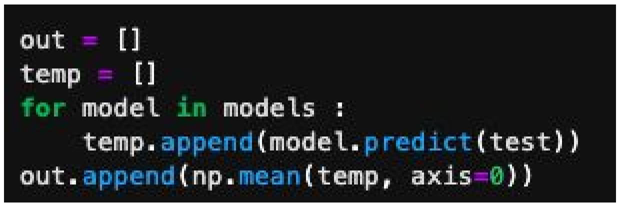


```
predict = np.zeros(len(train))
models = []
scores = []
X = np.arange(len(train))
ss = ShuffleSplit(n_splits=5, test_size=0.25, random_state=0)
for train_index,test_index in ss.split(X):
    xtr, ytr = train[features].iloc[train_index], train['Output'].iloc[train_index]
    xvl, yvl = train[features].iloc[test_index], train['Output'].iloc[test_index]
    regressor = LinearRegression()
    regressor.fit(xtr, ytr)
    models.append(regressor)
    predict[test_index]=(regressor.predict(xvl))


score = np.sqrt(metrics.mean_squared_error(train['Output'], np.clip(predict, a_min=0, a_max=None)))
```

I start by making an array called predict filled with 0s. I am using it to average out the different models I make by randomly dividing the data in order to cross-validate. I originally was using Stratified KFold to validate but I decided to go with a simpler randomized shuffle-split.

My average predicted score from cross validation was around 0.37, and when I tried random forest and LGBM I got similar results. I think my results might be better if I did more experimenting with what points are outliers. My fear is that I'm missing points that are outliers in some subdimensional space of the data, but this outlier-ness isn't reflected in the overall data because it's "more normal" in other dimensions. I wasn't really sure how to test for this though.



```
out = []
temp = []
for model in models :
    temp.append(model.predict(test))
out.append(np.mean(temp, axis=0))
```

I use an average of the models I made when cross-validating. I wish I had the time to experiment more within a linear space to see what weights of combinations of models would produce better results.

The hardest part was pre-processing and finding outliers. If the data was more concrete than just continuous variables with no column names I think pre-processing might have been easier, but maybe this was just randomly generated data.

When the instructions for the assignment said "You may want to use the same method/setting for datasets in folder a, and the same method/setting for datasets in folder b," I wasn't quite sure whether this was supposed to mean that we use the same models for all the data in folders 1, 2, 3, 4, and 5 within folder a, or whether it was recommended to generate separate models for each of these data sets, so I chose the latter. If I had more time, I would go back and see whether this was the case and whether this improved cross-validation scores or not.