**Background**

Jane works at a large-scale agricultural product processing and distribution plant. The company recently purchased a near-infrared spectroscopy (NIR) unit to determine the quality of several agricultural products. For example, NIR is used to estimate the brix level in peaches. The brix level is a measure of the sugar content in the peaches which is a key factor for judging peach quality.

**Problem**

The company receives peaches from many different suppliers. Customers of the plant have been complaining about the quality of peaches and related products. Jane shows you a sample of 50 NIR readings for the peaches along with their brix scores. The table containing NIR readings with corresponding brix values is 600 columns wide. Jane says they have not been able to build predictive models due to extreme multicollinearity. She complains, “My VIF scores are so high our computer printout says they are infinite!” You suggest using principal component analysis and provide Jane this starter code:

|  |
| --- |
| from sklearn.decomposition import PCA  from sklearn.metrics import mean\_squared\_error, r2\_score  from sklearn.linear\_model import LinearRegression  from sklearn.preprocessing import StandardScaler  import pandas as pd  import numpy as np  from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  from sklearn.model\_selection import KFold  # Read Brix value data.  PATH = "/Users/pm/Desktop/DayDocs/2019\_2020/PythonForDataAnalytics/workingData/"  CSV\_DATA = "peach\_spectra\_brixvalues.csv"  df = pd.read\_csv(PATH + CSV\_DATA, sep=',')  # Split the data using k-fold validation.  kfold = KFold(4, True) # 4-splits, shuffle randomly.  foldCount = 0  for train, test in kfold.split(df):  print("\n\n\*\*\*\*\*\*\* Fold Count: " + str(foldCount) + " \*\*\*\*\*\*\*")  foldCount += 1  # Extract train and test values.  dfTrain = df.iloc[train,:]  dfTest = df.iloc[test,:]  y\_train = dfTrain['Brix']  y\_test = dfTest['Brix']  dfXTrain = dfTrain.copy() # Copy to avoid affecting original df.  del dfXTrain['Brix'] # Drop Brix column.  dfXTest = dfTest.copy() # Copy to avoid affecting original df.  del dfXTest['Brix'] # Drop Brix column.  # Scale X values.  scaler = StandardScaler()  Xtrain\_scaled = scaler.fit\_transform(dfXTrain)  Xtest\_scaled = scaler.fit\_transform(dfXTest)  # Generate PCA components.  pca = PCA() # Adjust number of components with n\_components=  # Always fit PCA with train data. Then transform the train data.  X\_reduced\_train = pca.fit\_transform(Xtrain\_scaled)  # Transform test data with PCA  X\_reduced\_test = pca.transform(Xtest\_scaled)  print("\nPrincipal Components")  print(pca.components\_)  print("\nExplained variance: ")  print(pca.explained\_variance\_)  # Train regression model on training data  model = LinearRegression()  model.fit(X\_reduced\_train, y\_train)  # Predict with test data.  pred = model.predict(X\_reduced\_test)  # Show stats about the regression.  mse = mean\_squared\_error(y\_test, pred)  RMSE = np.sqrt(mse)  print("\nRMSE: " + str(RMSE))  '''  print("\nModel Coefficients")  print(model.coef\_)  print("\nModel Intercept")  print(model.intercept\_)  '''  from sklearn.metrics import r2\_score  print("\nr2\_score",r2\_score(y\_test,pred))  # For each principal component, calculate the VIF and save in dataframe  vif = pd.DataFrame()  # Show the VIF score for the principal components.  vif["VIF Factor"] = [variance\_inflation\_factor(X\_reduced\_train, i) \  for i in range(X\_reduced\_train.shape[1])]  print(vif["VIF Factor"]) |

Your task:

1. Display the count, min, max, percentile, standard deviation and average values for Brix.

|  |
| --- |
| Brix  count 50.000000  mean 16.436000  std 2.180504  min 11.200000  25% 15.150000  50% 16.750000  75% 18.100000  max 20.000000 |

1. In code, plot scree and cumulative variance graphs.

|  |
| --- |
|  |
|  |

1. Estimate the optimal number of components from the graphs.

The optimal number of components is 3 as the first three points wl1, wl2, and wl3 encompass a cumulative probability of roughly 90%

1. After selecting the optional number of components, in a table, show the RMSE, and explained variance for each of the principal components used in your model for each of the four folds.

|  |
| --- |
| Model 1  Principal Components  [[ 0.57735024 0.57735035 0.57735022]  [-0.67744993 -0.05598526 0.73343523]  [ 0.4557722 -0.81457487 0.35880285]]  Explained variance:  [3.08333234e+00 9.50040499e-07 4.23469558e-08]  RMSE: 2.9203481961128834  r2\_score -0.013091683389955255 |
| Model 2  Principal Components  [[ 0.57735025 0.57735033 0.57735023]  [-0.66415544 -0.07922889 0.74338438]  [-0.47493604 0.81264345 -0.33770754]]  Explained variance:  [3.08333255e+00 7.45162035e-07 3.77949553e-08]  RMSE: 2.094882354897291  r2\_score -1.3020110548927413 |
| Model 3  Principal Components  [[ 0.57735024 0.57735037 0.5773502 ]  [-0.66650324 -0.07519669 0.7417 ]  [-0.47163559 0.81302645 -0.34139106]]  Explained variance:  [3.08107994e+00 1.10601238e-06 3.49869707e-08]  RMSE: 1.5920239109731957  r2\_score 0.184361456258752 |
| Model 4  Principal Components  [[ 0.57735025 0.57735036 0.5773502 ]  [-0.64673331 -0.10826045 0.75499384]  [-0.49840016 0.80928749 -0.31088751]]  Explained variance:  [3.08108005e+00 1.00186835e-06 2.83050792e-08]  RMSE: 2.1010582591501183  r2\_score -0.11695283314833671 |

1. Show me your table with a sample of one scree plot and one cumulative variance plot before the end of class.

You can work with others and you can discuss your results. You may also view each other’s work over the shoulder but do not share your work electronically by any means.

Note: You will notice that will appear as negative in the printout. While we have calculated using , it is possible to obtain values by other methods. Other formulas can lead to negative results which means the model is out-performed by a baseline guess which just uses the average brix level. In other words, negative means the model makes no sense.

For this example, you should be able to discover promising potential for developing a decent linear regression model. In spite of some negative results during certain folds, you can suggest to Jane that the results will become more reliable once more data can be obtained for training and testing.

References:

<https://nirpyresearch.com/principal-component-regression-python/>