IE7429 Term Project

Machine Learning:

Linear regression

Cheyenne Siuba

Steve Austin

- **Introduction**:

This project is developed to explore machine leaning and its practical applications for industrial engineers. We were provided a dataset describing measurements of contaminants found in the cooling oil used in power transformers used in the electrical transmission grid. The health index column provides the desired output. Our task was to apply machine learning concepts to this scenario to provide a continuous output prediction model for future oil samples taken from power transformers. We used a linear regression model to compare correlations of the possible contaminants to the health index. Our model is from the "sklearn" python library and is appropriately labeled "Linear regression".

Our data was given in 471 rows (470 data) and 15 columns (14 independent variables) The columns consisted of 14 potential risk factors toward the Health index. The Health index relates to a matrix that gives life expectancy of the transformer and maintenance requirements to support the equipment as it ages. The columns also listed 10 oil contaminants that are detected in samples. The presence and quantity of the contaminants may help predict the transformers Health index. The contaminants are hydrogen, oxygen, nitrogen, methane, CO, CO2, ethylene, ethane, acetylene, and water content. The other possible predictive features of the transformer's health are an oil life additive, DBDS, the interfacial strength and dielectric rigidity of the oil, and las the transformers power factor.

- **Preprocessing** & **Feature analysis**

Our data set was straight forward and did not require adding, removing or other data manipulation techniques to prepare the data. We identified two feature labels with typos and corrected them accordingly as part of preprocessing. Our understanding of the given features led us to assume that all the data features potentially affect the Health Index. We combined feature analysis as part of our preprocessing step because it is the best way to prepare this specific data set for machine learning.

From the histograms, the data set appears to have high correlations between several features and with the output. Fields that have similar correlations can distort and hinder the effectiveness of machine learning algorithms, so we will find the most likely Health Index determinant variables through use of a correlation matrix and a field selection algorithm. Both feature selection techniques use correlation techniques, so we expect similar results from both tests. Running both test is a means to proof and rely on the results of the two feature selections methods.

The histograms also indicated a large disparity between the ranges amongst the features. Machine learning is more effective when the ranges are in the same scale. To adjust our dataset to a machine learning efficient data frame, we'll scale the features with normalization. Since our feature selection algorithm is also a form of machine learning, we'll normalize the data first.

    - Include with your discussion the Python code used for this task.

Normalization Code:

```
dfNoHI = df.loc[:]
col_names =  dfNoHI.columns
dfNoHI[col_names] =
pd.DataFrame(pre.StandardScaler().fit_transform(pd.DataFrame(dfNoHI[col_names])))
df = dfNoHI
```

Correlation Matrix Code:

```
#Correlation Matrix
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['figure.figsize'] = 14, 7
rcParams['axes.spines.top'] = False
rcParams['axes.spines.right'] = False

#create correlation matrix
corrM = df.corr()

#stregnth of correlations
out_HI=corrM[["Health index"]]
out_HI=out_HI.apply(abs)
out_HI.sort_values(by="Health index",inplace=True, ascending=False)
print(out_HI)
```

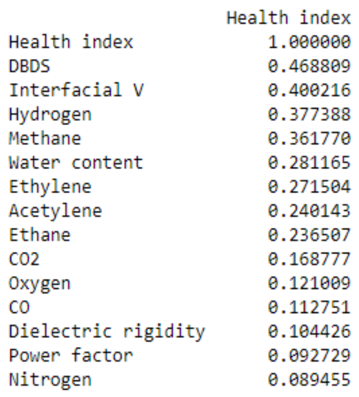
Feature Selection Algorithm Code:

```
X = df.drop(columns=['Health index'])
Y = df['Health index']

X = X.astype(int)
Y = Y.astype(int)


#Train the classifier.
from xgboost import XGBClassifier
model = XGBClassifier()
```

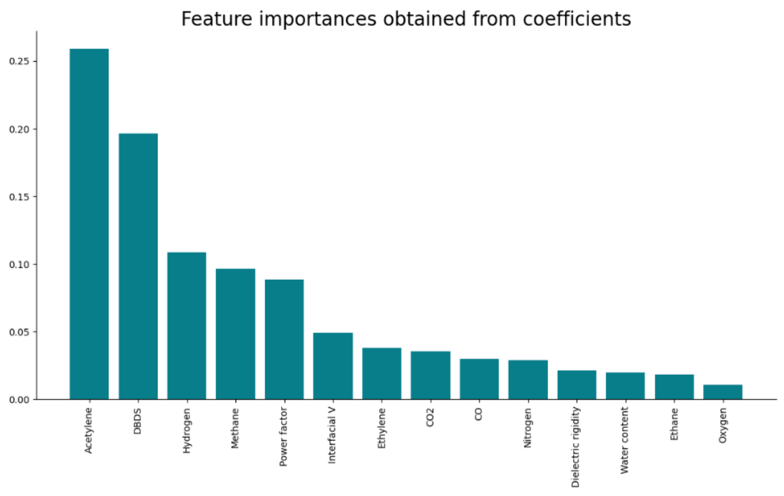
```
model.fit(X,Y)

#Extract importance for each feature (Attribute) from trained model
importances = pd.DataFrame(data={
   'Attribute': X.columns,
   'Importance': model.feature_importances_})

#sort by importance in descending order, and plot
importances = importances.sort_values(by='Importance', ascending=False)
plt.bar(x=importances['Attribute'], height=importances['Importance'], color='#087E8B')
plt.title('Feature importances obtained from coefficients', size=20)
plt.xticks(rotation='vertical')
plt.show()
```

Comparative Analysis:

The correlation matrix and the feature selection algorithm are agreeable, so we will drop their lowest correlated fields from the machine learning algorithm for Health index.



```
                     Health index
Health index             1.000000
DBDS                     0.468809
Interfacial V            0.400216
Hydrogen                 0.377388
Methane                  0.361770
Water content            0.281165
Ethylene                 0.271504
Acetylene                0.240143
Ethane                   0.236507
CO2                      0.168777
Oxygen                   0.121009
CO                       0.112751
Dielectric rigidity      0.104426
Power factor             0.092729
Nitrogen                 0.089455
```



Selected Features:

Health index, DBDS, Interfacial V, Hydrogen, Methane, Water content, Ethylene, Acetylene, Ethane, CO2, Oxygen

Results:

We execute the linear regression machine learning library from sklearn. Training was performed by first separating a training set and testing set from the provided data. We used a 20% sample selection for the test set and used the other 80% to train our model. Our coding then divided our model into dependent, Health index, and independent variables, all other features less those identified by our feature selection process.

- Evaluation

Our final product produced an R^2 of 0.507 whereas our initial test with all features gave us R^2 of 0.517. However, our differential statistics between the two tests were considerably different. Our initial test consisted of raw data to better understand the effects of our preprocessing and feature selection methods. While we expected to see significant improvement in the R^2 value from feature selection, it appears that we may need to further restrict our feature filter to achieve that result. The most noticeable changes of central tendency are likely due to normalizing our data for the final test. Further testing and analysis is required to confirm our findings.

	Final Test	Initial Test
R^2	0.507	0.517
count	94.000000	118.000000
mean	0.512549	8.907487
std	0.372977	6.915288
min	0.002525	0.047591
25%	0.257657	3.983060
50%	0.431484	6.551967
75%	0.834646	14.576568
max	1.830341	32.129101

Linear Regression Machine Learning Code:

```
dfReg = df.drop(columns=['Dielectric rigidity', 'Power factor','Nitrogen'])

from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
train = dfReg.sample(frac=0.80, random_state=125)
test = dfReg.drop(train.index)
x = train.drop(columns=['Health index'])
y = train[["Health index"]]
x = sm.add_constant(x)
model = sm.OLS(y,x).fit()

xtest = test.drop(columns=['Health index'])
xtest = sm.add_constant(xtest)
ytest = test[["Health index"]]
model.summary()
ypredictions=pd.DataFrame()
ypredictions["prediction"] = model.predict(xtest)
ycompare=pd.merge(ytest, ypredictions, left_index=True, right_index=True)
ycompare.head()
ycompare["difference"]=abs(ycompare["Health index"]-ycompare["prediction"])
ycompare["difference"].describe()
```