“Use of Linear Regression to Determine Risk Metric for Operation Failure under Current Conditions on 3 CNC Machines.”

Context and Problem Statement:

A company is using three CNC machines with almost identical capabilities for machining operations on a production line. These machines are located near the stamping operation which generates a considerable amount of vibration when in use. The current operation CNC operation structure is scheduled to run 2 shifts per day shift A and B. The operations team wants to understand which variable (measurement of environmental feature) or combination of variables influence the most or lead to a quality rejection on the final quality check at the end of their process.

Introduction Method:

For this project we developed a regression model, in general this model searches relationships amongst variables, and it’s started by selecting a phenomenon of interest in this case the risk of machine failure and capturing a certain number of observations (3410 in our case) of at least 2 features (8 in this problem), where the assumption is that there’s a dependency correlation where at least one of the features depends on the others and their interaction. This model is based on Linear Regression, one of the most widely used regression methods and that assumes that the relationship between the one dependent variable (quality decision, good/fail) and one or more independent variables (time of the day, machine, room temperature, power, peak to peak vibration levels and machine wear) is a linear function.

Dataset:

For this purpose, the team has captured 3410 raw samples including the following information each, column 1: time of day is in fractional hours since midnight, column 2: Machine number (1-3), column 3: Room temperature is in degrees F, column 4: Power, a metric based on power draw (machines having issues often have higher power draw), column 5: peak-to-peak measure “ambient” vibration levels at each machine (higher value indicates more vibration), column 6: Wear Measure is a metric of tool wear (higher indicates more wear), and column 7: which has the final quality decision (good/fail or output) for each part.

Data Preprocessing:

Before a regression model can be ran there are multiple data preprocessing steps that need to be completed for quality insurance for our model. These steps included:

1. Initial Feature selection: based on the desired output of our model which is aimed to determine the dependency correlation of environmental features with the final quality check decision, that’s why the latter is assigned as the output variable while all the other ones will be the attributes or input variables.
2. Defining Data source: the samples used in our model were collected in an excel spreadsheet, which due to their availability form (rectangular), is considered as a structured data set. One of our natural first steps was to import the dataset into python where the model was developed.

Python code used:

df = pd.read\_excel("C:/Users/yvoar/Desktop/courses/Data Analytics/term project/QC Dataset.xlsx")

df.head()

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.max\_rows', None)

1. Next, we performed a data characterization to understand what type of data we had on each column, obtaining the following results were “float64” and “int64” both refer to numerical values, while “object” found only in the shift column refers to a categorical value.

Python code used:

print('Basic info about the data frame and its rows and columns:')

print(df.info ())

Output:

Text

Description automatically generated

1. Followed by a count of null of missing values to understand if we had gaps in our dataset, in the results we can observe that there weren’t any null or missing values in the data set.

Python code used:

print("Number of null/missing values in each column:")

print(df.isna().sum())

Output:

Text

Description automatically generated

1. We proceeded to encode the non-numerical data which in this case was only present on the “Shift” column, therefore we replaced all Shift “A” entries for 1 and all Shift “B for 2.

Python code used:

df["Shift"].replace("A",1,inplace= True)

df["Shift"].replace("B",2,inplace= True)

1. We then performed a count of “unique values to understand how many different values we had on each of the column, which other than “shift” and “QC” results that are binary columns, and “machine” that is ordinal since you can only chose one of the 3 machines available which makes it an ordinal column, but for the most part all the other numerical columns have a large number of different data points.

Python code used:

print("Number of unique values for all columns:")

print(df.nunique(axis=0))

Output:

Text

Description automatically generated

1. Subsequently we ran basic and graphical statistics and for all numerical columns to understand observe specially on the numerical values how they behave, from this we observed how some numerical values could present skewness problems as their mean are median are quite distant from each other this is true for numerical columns of “Power”, “Peak to Peak” and “Wear Measure”.

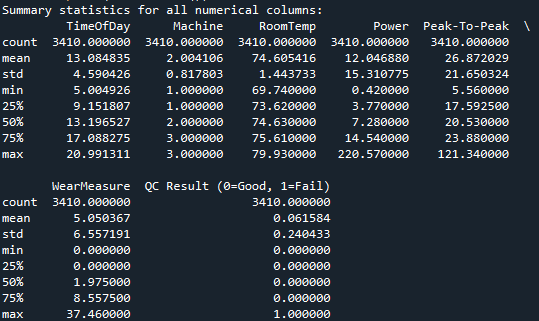
Python code used:

print('Summary statistics for all numerical columns:')

print(df.describe())

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.max\_rows', None)



And we Ran some graphical tools to validate:

The histogram and distplot charts below illustrate the skewness of the numerical columns mentioned priorly, followed by the boxplot graphic that once again lets us see that in these columns a large amount of the values getting marked as outliers due to this problem.

* Histogram: Python code: print("Duplicate rows:")

print(df[df.duplicated(keep=False)])

df.hist()

A picture containing diagram

Description automatically generated

* Distplot: Python code: fig, ax = plt.subplots(ncols=4, nrows=2, figsize=(20, 10))

ndex = 0

ax = ax.flatten()

for col, value in df.items():

sns.distplot(value, ax=ax[index])

index += 1

plt.tight\_layout(pad=0.5, w\_pad=0.7, h\_pad=5.0)

Chart, histogram

Description automatically generated

* Boxplot: Python code:

fig, ax = plt.subplots(ncols=4, nrows=2, figsize=(20, 10))

index = 0

ax = ax.flatten()

for col, value in df.items():

sns.boxplot(y=col, data=df, ax=ax[index])

index += 1

plt.tight\_layout(pad=0.5, w\_pad=0.7, h\_pad=5.0)

Chart, box and whisker chart

Description automatically generated

1. The next step was to fix the skewed data columns (“power’, “peak to peak”, “wear measure”) so these wouldn’t interfere with the machine learning model developed later, and scale all the remaining numerical columns (“room temp” and “time of the day”) and run the graphical tools again to validate this step.

Python code used for these steps:

Transform data:

qt = pre.QuantileTransformer(output\_distribution='normal')

df['Power'] = pd.DataFrame(qt.fit\_transform(pd.DataFrame(df['Power'])))

qt = pre.QuantileTransformer(output\_distribution='normal')

df['Peak-To-Peak'] = pd.DataFrame(qt.fit\_transform(pd.DataFrame(df['Peak-To-Peak'])))

qt = pre.QuantileTransformer(output\_distribution='normal')

df['WearMeasure'] = pd.DataFrame(qt.fit\_transform(pd.DataFrame(df['WearMeasure'])))

Scale remaining columns:

col\_names = ['RoomTemp', 'TimeOfDay'] #numeric column names to scale

df[col\_names] = pd.DataFrame(pre.StandardScaler().fit\_transform(pd.DataFrame(df[col\_names])))

Graphics after step:

Histogram:

Chart

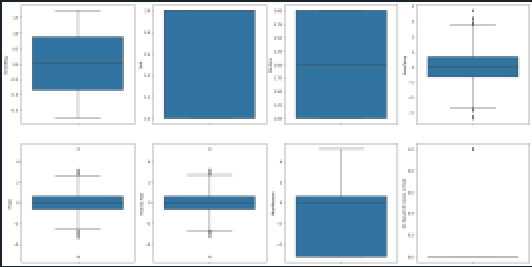
Description automatically generated with low confidence

Distplot:

Histogram

Description automatically generated

Boxplot:



Feature Analysis:

Once the preprocessing or data treatment steps was completed, in this step we analyzed all the columns or features within our sample to understand which ones correlate to our output column of “QC result”, and decide which ones were the most relevant to keep in our training model. Based on the results of this analysis we were able to tell that our features were not as highly correlated with the output, except for “power” and “wear measure” that were somewhat positively correlated meaning that as these increased the chances of getting “1” (bad part) as a final QC result also increased. As a final decision we decided to remove at least the two least correlated columns that were Machine and peak to peak as shown in the graphics below.

All features correlation matrix: Python code: corrM = df.corr()

A screenshot of a computer

Description automatically generated with medium confidence

Ranking correlation based on output metric “QC result”: Python code: out\_class=corrM[["QC Result (0=Good, 1=Fail)"]]

out\_class=out\_class.apply(abs)

out\_class.sort\_values(by="QC Result (0=Good, 1=Fail)",inplace=True, ascending=False)

Graphical user interface

Description automatically generated with medium confidence

Afterwards and in preparation for our training model, we proceeded to split the data in input and output features and remove the two least correlated columns previously mentioned.

Python code: x = df.drop(columns=['QC Result (0=Good, 1=Fail)','Peak-To-Peak','Machine'])

y = df['QC Result (0=Good, 1=Fail)']

Machine Learning model

With all the pre-steps completed it was then time to train and assess our model, for which we imported the needed tools from “sklearn”, define our training model paradigm, the prediction of the training set and perform the cross validation, we decided to use 2 main tools or metrics to assess the “goodness of fit” or cross validate our regression model, the first one R2 score (the proportion of the variance in the dependent variable that is predictable from the independent variable(s)), the mean square error or MSE (the average of the square of the errors. The larger the number the larger the error. Error in this case means the difference between the observed values y1, y2, y3, … and the predicted ones pred(y1), pred(y2), pred(y3)).by using the following python code

* Model training import packages:

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

def train(model, x, y):

* train the model

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=42)

model.fit(x\_train, y\_train)

* perform cross-validation

cv\_score = cross\_val\_score(model, x, y, scoring='neg\_mean\_squared\_error', cv=5)

cv\_score = np.abs(np.mean(cv\_score))

print("Model Report")

print("MSE:",mean\_squared\_error(y\_test, pred))

print('CV Score:', cv\_score)

print("R2 score : %.2f" % r2\_score(y\_test,pred)

Then we ran the regression with the different tools to decide which pose a better fit based on the output, in this case we got a “feature importance ranking” from each tool and the R2 and MSE. The regression tools we included in this step from “sklearn” were linear regression, decision tree regressor and random forest regressor the results and python code for this task were as follows.

Linear regression Python code:

from sklearn.linear\_model import LinearRegression

model = LinearRegression(normalize=True)

train(model, x, y)

coef = pd.Series(model.coef\_, x.columns).sort\_values()

coef.plot(kind='bar', title='Model Coefficients')

Feature importance ranking:

Chart

Description automatically generated

MSE: 0.04

R2 score: .20 or 20%

Decision Tree Regressor Python code:

from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor()

train(model, x, y)

coef = pd.Series(model.feature\_importances\_, x.columns).sort\_values(ascending=False)

coef.plot(kind='bar', title='Feature Importance')

Feature importance ranking:

Chart, bar chart

Description automatically generated

MSE: .01

R2 Score: .75 or 75%

Random Forest Regressor Python code:

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()

train(model, x, y)

coef = pd.Series(model.feature\_importances\_, x.columns).sort\_values(ascending=False)

coef.plot(kind='bar', title='Feature Importance')

Feature importance ranking:

Chart

Description automatically generated

MSE: .007

R2: 0.88 or 88%

Conclusion

After running our machine learning model with all 3 different regression models we see a difference specially with the first model were the results for ranking feature by importance that although places power one of the most highly correlated features with respect to the out class but moves another more correlated feature to a third place in the ranking but most importantly when we look at our model validity indicators of MSE and R2 (where we want to keep our Error results as low as possible and our R2 as high as possible) this model has the lowest performance and the next two models have a better chance of working well in practice, based on this parameters although is close with the Decision tree Regressor, the model we chose and that has the highest chance of success in this task of working well in practice and be sufficient in predicting or explaining our output class results is the one with Random forest regression.