Feature Creation Deliverable

In this notebook the clean data from the ETL notebook is transformed into data for the final model. The machine_status column is transformed from NORMAL, RECOVERING, and BROKEN into 1,0 with BROKEN being combined into RECOVERING. Timestamp column of the DataFrame is transformed into datetime from object type. Finally the sensor data is scaled using StandardScaler to correct the data back into the order, i.e sensor A = 2 vs sensor B = 200.

The following block of code is where the project information is read into the notebook for saving the final results of the Feature Creation. The block of code following the first hidden cell is where data_clean data is read into the notebook from the IBM Cloud Object Store.

In [29]:
```
# The code was removed by Watson Studio for sharing.
```

In [30]:
```
# The code was removed by Watson Studio for sharing.
```

Out[30]:

	Id	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08	sensor_09	sensor_10	s
0	0	2018-04-01 00:00:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	37.22740	
1	1	2018-04-01 00:01:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	37.22740	
2	2	2018-04-01 00:02:00	2.444734	47.35243	53.2118	46.397570	638.8889	73.54598	13.32465	16.03733	15.61777	15.01013	37.86777	
3	3	2018-04-01 00:03:00	2.460474	47.09201	53.1684	46.397568	628.1250	76.98898	13.31742	16.24711	15.69734	15.08247	38.57977	
4	4	2018-04-01 00:04:00	2.445718	47.13541	53.2118	46.397568	636.4583	76.58897	13.35359	16.21094	15.69734	15.08247	39.48939	

In [31]:
```
#This is where the timestamp data is transformed into a datetime type instead of the object type from the cloud
data_clean['timestamp'] = [datetime.strptime(x, '%Y-%m-%d %H:%M:%S') for x in data_clean.timestamp]
data_clean.head()
```

Out[31]:

	Id	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08	sensor_09	sensor_10	s
0	0	2018-04-01 00:00:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	37.22740	
1	1	2018-04-01 00:01:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	37.22740	
2	2	2018-04-01 00:02:00	2.444734	47.35243	53.2118	46.397570	638.8889	73.54598	13.32465	16.03733	15.61777	15.01013	37.86777	
3	3	2018-04-01 00:03:00	2.460474	47.09201	53.1684	46.397568	628.1250	76.98898	13.31742	16.24711	15.69734	15.08247	38.57977	
4	4	2018-04-01 00:04:00	2.445718	47.13541	53.2118	46.397568	636.4583	76.58897	13.35359	16.21094	15.69734	15.08247	39.48939	

In [32]:
```
#Any machine_status of BROKEN is transformed into a RECOVERING stage. This is because the goal is to test for r
#Since BROKEN and RECOVERING is both options of non-normal they are both rolled into RECOVERING
data_clean.loc[data_clean.machine_status == 'BROKEN', 'machine_status'] = 'RECOVERING'
data_clean.machine_status.value_counts()
```

Out[32]:
```
NORMAL        205836
RECOVERING     14484
Name: machine_status, dtype: int64
```

In [33]:
```
#In this code cell the data is transformed from categorical data to numerical data.
#1:1 and 0:0 are in the dictionary for if any repeated running of this code block causes Nans to appear instead
data_clean['machine_status'] = data_clean.machine_status.map({'NORMAL': 1, 'RECOVERING': 0, 1:1, 0:0})
data_clean.machine_status.value_counts()
```

Out[33]:
```
1    205836
0     14484
Name: machine_status, dtype: int64
```

In [34]:
```
#StandardScalar is used here to correct all of the data into the proper a -1 to 1 value for each sensor
from sklearn.preprocessing import StandardScaler

sensor_data = data_clean.loc[:,'sensor_00':'sensor_51']
clean_col = data_clean.columns[2:-1]
data_clean[clean_col.values] = StandardScaler().fit_transform(X = sensor_data)
data_norm = pd.DataFrame(data_clean)
data_norm.head()
```

Out[34]:

	Id	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08	sensor_09	sensor_10	s
0	0	2018-04-01 00:00:00	0.231450	-0.151675	0.639386	1.057675	0.303443	0.177097	-0.042091	0.132586	0.181964	0.122858	-0.350860	
1	1	2018-04-01 00:01:00	0.231450	-0.151675	0.639386	1.057675	0.303443	0.177097	-0.042091	0.132586	0.181964	0.122858	-0.350860	
2	2	2018-04-01 00:02:00	0.180129	-0.072613	0.639386	1.093565	0.334786	0.008647	-0.082656	0.089329	0.207112	0.101892	-0.297906	
3	3	2018-04-01 00:03:00	0.219228	-0.151675	0.627550	1.093564	0.260045	0.207693	-0.086035	0.185835	0.246628	0.136839	-0.239029	
4	4	2018-04-01 00:04:00	0.182573	-0.138499	0.639386	1.093564	0.317909	0.184568	-0.069133	0.169195	0.246628	0.136839	-0.163810	

In [8]:
```
#Final part of the Feature Creation Deliverable where the data is saved to object store.
project.save_data(file_name = 'final_data.csv', data = data_norm.to_csv(index=False), overwrite=True)
```

Out[8]:
```
{'file_name': 'final_data.csv',
 'message': 'File saved to project storage.',
 'bucket_name': 'fundamentalsofscalabledatascience-donotdelete-pr-9lqqxd4zzrrymc',
 'asset_id': '8fc7ecf8-6eb5-4363-bff2-a305d61a4af0'}
```

One Additional Iteration in Feature Creation

For the additional feature creation step PCA Decomposition is done to the data. Decomposing this data makes sense as over 50 sensors is a large amount of dimensions and PCA can help to reduce the time for the models to train.

In [35]:
```
from sklearn.decomposition import PCA

#Separate the sensor dimensions from the machine_status
X = data_norm.loc[:,'sensor_00':'sensor_51']
y = data_norm.machine_status

#PCA Decomposition
pca = PCA(n_components=25).fit_transform(X)
```

In [38]:
```
#Turn numpy array to DataFrame
X_pca = pd.DataFrame(pca)

#The machine_status is joined to the PCA X
final_pca = X_pca.join(y)
```

In [40]:
```
#One Additional Step in Feature Creation is saved to Object Store
project.save_data(file_name = 'final_pca.csv', data = final_pca.to_csv(index=False), overwrite=True)
```

Out[40]:
```
{'file_name': 'final_pca.csv',
 'message': 'File saved to project storage.',
 'bucket_name': 'fundamentalsofscalabledatascience-donotdelete-pr-9lqqxd4zzrrymc',
 'asset_id': 'cdeaf2ad-34f1-4f3b-9611-3a57bc061a48'}
```