Jacob Blumsack

COMP 4220 Machine Learning Final: Regression Set

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
#Load dataset
concreteData = pd.read_csv("Concrete_Data.csv")
#display set
concreteData.head()
```

Out[2]:

	Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5) (kg in a m^3 mixture)	Coarse Aggregate (component 6)(kg in a m^3 mixture)	Fine Aggregate (component 7)(kg in a m^3 mixture)
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5

In [345]:

```
#Now we want to split the data into training and testing sets
from sklearn.model_selection import train_test_split

#set the X and y setvs
X = concreteData.drop('Concrete compressive strength(MPa, megapascals)
 ', axis = 1)
y = concreteData['Concrete compressive strength(MPa, megapascals) ']

#use train test split for test and train sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

In [4]:

```
#Now create the linear regression object
from sklearn import linear_model

#the linear model is created
lm = linear_model.LinearRegression()

#the model is trained
lm.fit(X_train, y_train)

#then make the predictions with the test set
y_pred = lm.predict(X_test)
print(y_pred)
```

```
[39.54173004 14.34766636 61.33148723 54.02870482 24.44257981 53.6309445
9
 45.98973566 27.31187439 53.23177832 37.08298577 16.75983422 39.4324516
 29.69334674 35.87961914 47.63884285 56.70862412 35.79871479 29.3630872
5
 48.69052196 35.33194214 53.80404465 32.10561488 33.046423   48.2397051
6
 23.61819651 23.53591425 69.42292047 26.76815501 56.70862412 48.2333889
1
 18.9707685  36.43996199 17.42660579 19.52438719 24.61010586 15.2843709
 53.18012778 28.15463908 27.19220514 26.32272738 52.53006928 30.4211548
1
 25.78127338 35.57455811 53.778552   53.99954893 30.44100048 44.1461646
4
```

```
 27.16386707 57.07814289 36.70330382 19.1206505  23.22652397 34.6523315
9
 43.68926059 47.45581253 35.93488017 53.14095914 59.97287158 36.6441340
4
 31.15430767 20.63766861 50.81476739 55.59642861 20.67748719 44.2723334
5
 33.26486213 34.19218083 19.36508085 20.12460607 28.50014411 27.9053554
1
 54.45036573 19.52883535 36.53336599 50.73530147 28.2510113  19.2110382
7
 23.92475387 28.13662921 18.12865721 44.69057698 33.11727514 33.2171859
 38.48765977 34.82395079 49.01036872 50.64832702 38.79295445 31.5017314
6
 52.94685815 20.55710969 26.07212818 37.3765736  26.46988824 55.7361718
5
 32.75695309 33.82598965 51.50390997 27.0804715  30.35479724 38.4449798
4
 25.37124505 32.01080185 34.25274625 35.35396881 39.53654851 32.6526583
3
 28.10106    39.71562193 18.3383156  48.92165464 25.40018595 33.1914939
2
 40.04244436 64.09480821 31.29589063 35.47632435 41.05579242 31.9517477
1
 42.35933507 25.83166994 60.94904769 26.53762912 17.94215958 20.2321103
2
 15.80039131 19.23096169 40.11868644 32.57083932 30.17452876 47.2263638
6
 33.75408333 24.46999351 19.35613166 54.53287457 61.41481008 36.6949301
2
 11.3561288  18.7891324  38.2658533  46.79880383 32.86190911 17.7899975
2
 16.71311028 35.72559271 48.2815388  25.43305003 27.48753363 16.6431715
4
 24.86054177 28.38943703 26.67315248 19.09682716 31.01511406 30.7225637
6
 47.940352   30.38164541 30.6973179  22.37542679 38.99061925 42.3061123
1
 23.85396383 25.98498473 33.04462783 23.2511946  56.22302284 53.5529064
9
 57.50570292 11.07744576 30.86133857 33.66699176 24.51156194 58.8531333
```

```
9
 21.73012463 21.6715475  16.76429778 48.96693486 23.43349435 39.5837414
7
 33.1508203  31.80351525 19.05087087 39.01530395 31.12162819 27.5261069
6
 40.5645329  39.77782127 28.2794236  50.81476739 21.13481475 72.5786381
 44.40289593 27.85947812 14.5952874  29.01630375 31.45197306 27.5991507
1
 30.83386442 58.50339193 17.54279168 20.87181766 41.32846477 31.6582901
4
 33.38443905 23.72002409 39.03462342 30.55013048 30.96060592 25.3795387
9
 49.34699553 22.91129371 23.1092617  22.65157583 26.08448762 21.7875450
5
 22.47657844 39.31632164 19.55695594 39.96129584 57.60243918 31.8301572
 38.86583991 36.42802384 41.33046758 36.12171674 25.89789821 32.6426614
 34.30013127 45.38591085 36.45361983 37.13526249 61.17733737 21.2782560
3
 35.44941567 31.80667679 31.65204539 36.22052157 31.00588996 59.5417716
2
 59.15345043 26.93960768 42.88523636 33.06990341 27.90908496 31.3776967
2
 49.13712175 28.20729783 35.20545144 15.77694914 11.66984552 40.0558742
1
 35.98386571 31.06107304 38.74201295 21.55796565 24.54719309 19.5976874
6]
```

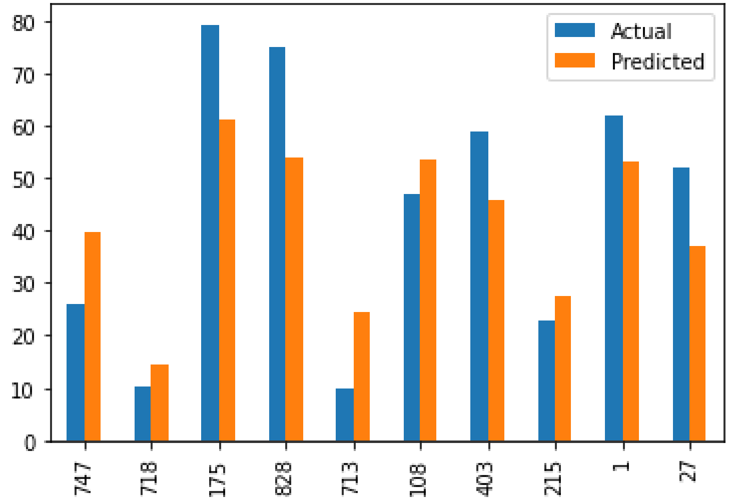
In [5]:

```
#calculate the metrics
from sklearn import metrics

accuracy = lm.score(X_test, y_test)
meanAbsoluteError = metrics.mean_absolute_error(y_test, y_pred)
meanSquaredError = metrics.mean_squared_error(y_test, y_pred)

print('Accuracy is', accuracy)
print('Mean Absolute Error is', meanAbsoluteError)
print('Mean Squared Error is', meanSquaredError)
```

```
Accuracy is 0.6234704748356753
```

```
Mean Absolute Error is 7.780889898045493
Mean Squared Error is 95.88863777582235
```

In [6]:
```
#we can compare the results predicted vs the actuals
concretePlot = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

#we will just sample the first 10
dat1 = concretePlot.head(10)
dat1.plot(kind = 'bar')
plt.show()
```



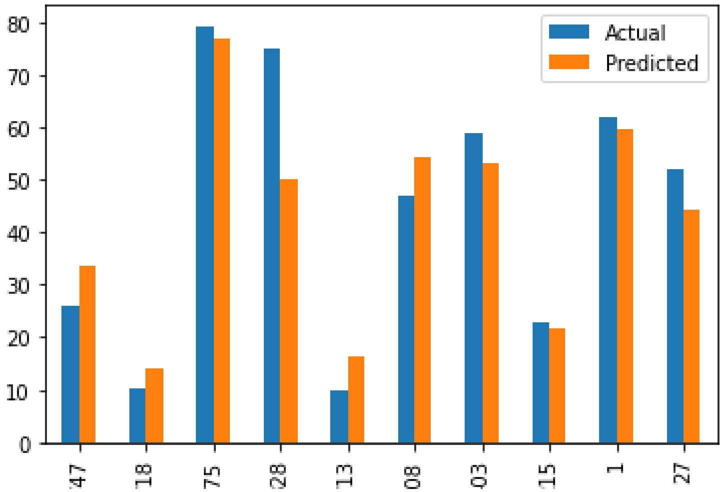
In [7]:
```
#Now lets compare this to Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures

#we will use degree 2
polyReg =  PolynomialFeatures(degree = 2)

#now fit the model
XTrain_polyReg = polyReg.fit_transform(X_train)
XTest_polyReg = polyReg.fit_transform(X_test)

#now for the new prediction
polyReg_lm = linear_model.LinearRegression()
```

```python
polyReg_Model = polyReg_lm.fit(XTrain_polyReg, y_train)
polyReg_pred = polyReg_lm.predict(XTest_polyReg)
```

In [9]:
```python
#calculate the metrics
accuracy = polyReg_Model.score(XTest_polyReg, y_test)
meanAbsoluteError = metrics.mean_absolute_error(y_test, polyReg_pred)
meanSquaredError = metrics.mean_squared_error(y_test, polyReg_pred)

print('Accuracy is', accuracy)
print('Mean Absolute Error is', meanAbsoluteError)
print('Mean Squared Error is', meanSquaredError)
```

```
Accuracy is 0.7241629175542621
Mean Absolute Error is 6.302540294588074
Mean Squared Error is 70.24586470937672
```

In [10]:
```python
#we can compare the results predicted vs the actuals
concretePlot_poly = pd.DataFrame({'Actual': y_test, 'Predicted': polyRe
g_pred})

#we will just sample the first 10
dat1 = concretePlot_poly.head(10)
dat1.plot(kind = 'bar')
plt.show()
```



7 7 1 8 7 1 4 2

In [11]:
```
#Now lets compare to Artificial Neural Network
#first get the needed libraries
import tensorflow as tf
from sklearn.compose import ColumnTransformer
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [336]:
```
#the data must be split and preprocessed, this time with min max scaler
from sklearn.preprocessing import MinMaxScaler
minMax = MinMaxScaler()
X_train = minMax.fit_transform(X_train)
X_test = minMax.transform(X_test)

#Time to make the ANN
classifier = Sequential()
```

In [364]:
```
#add the imput layer and first hidden layer
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activat
ion = 'relu', input_dim = 8))
```

In [365]:
```
#now add the second hidden layer
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activat
ion = 'relu'))
```

In [366]:
```
#finally add the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activat
ion = 'sigmoid'))
```

In [367]:
```
#time to compile it
classifier.compile(optimizer = 'Adam', loss = 'binary_crossentropy', me
trics = ['accuracy'])
classifier.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_103 (Dense)            (None, 6)                 54
_________________________________________________________________
dense_104 (Dense)            (None, 6)                 42
_________________________________________________________________
dense_105 (Dense)            (None, 1)                 7
_________________________________________________________________
dense_106 (Dense)            (None, 6)                 12
_________________________________________________________________
dense_107 (Dense)            (None, 6)                 42
_________________________________________________________________
dense_108 (Dense)            (None, 1)                 7
_________________________________________________________________
dense_109 (Dense)            (None, 6)                 12
_________________________________________________________________
dense_110 (Dense)            (None, 6)                 42
_________________________________________________________________
dense_111 (Dense)            (None, 1)                 7
_________________________________________________________________
dense_112 (Dense)            (None, 6)                 12
_________________________________________________________________
dense_113 (Dense)            (None, 6)                 42
_________________________________________________________________
dense_114 (Dense)            (None, 1)                 7
=================================================================
Total params: 286
Trainable params: 286
Non-trainable params: 0
_________________________________________________________________
```

In [368]:

```
#now the data comes in, the ANN is fit to the training set
classifier.fit(X_train, y_train)
```

```
25/25 [==============================] - 1s 1ms/step - loss: 0.4715 - a
ccuracy: 0.0000e+00
```

Out[368]:

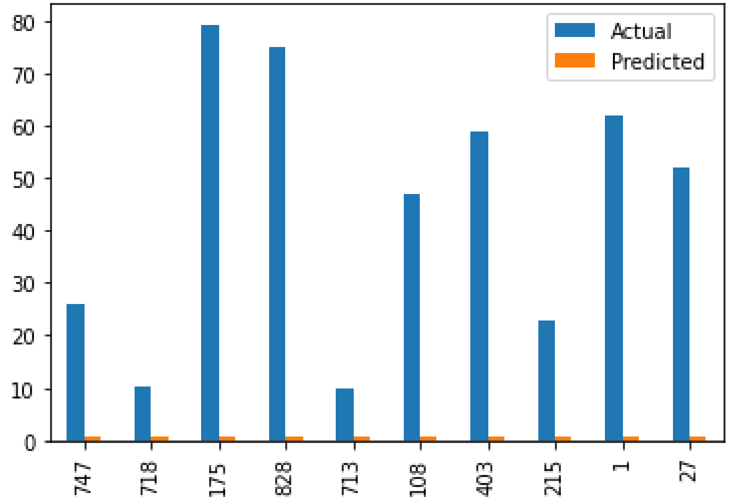
```
<tensorflow.python.keras.callbacks.History at 0x1f6341d6550>
```

In [369]:
```
#now predict the test set results
Pred = classifier.predict(X_test)
#Pred = (Pred > 0.1)
```

In [370]:
```
#calculate the metrics
meanAbsoluteError = metrics.mean_absolute_error(y_test, Pred)
meanSquaredError = metrics.mean_squared_error(y_test, Pred)

print('Mean Absolute Error is', meanAbsoluteError)
print('Mean Squared Error is', meanSquaredError)
```

```
Mean Absolute Error is 34.024621150105496
Mean Squared Error is 1412.339170489872
```

In [371]:
```
#we can compare the results predicted vs the actuals
concretePlot_ANN = pd.DataFrame({'Actual': y_test, 'Predicted': Pred.fl
atten()})

#we will just sample the first 10
dat2 = concretePlot_ANN.head(10)
dat2.plot(kind = 'bar')
plt.show()
```



```
In [ ]:
```