January 30, 2018

Lab Assignment 1

Sanchit Singhal

INF 385T – Introduction to Machine Learning with Danna Gurari

Spring 2018

School of Information

Hyperlink to code: https://introtoml-sanchit1276.notebooks.azure.com/nb/notebooks/IntroToML/LabAssignment1.ipynb

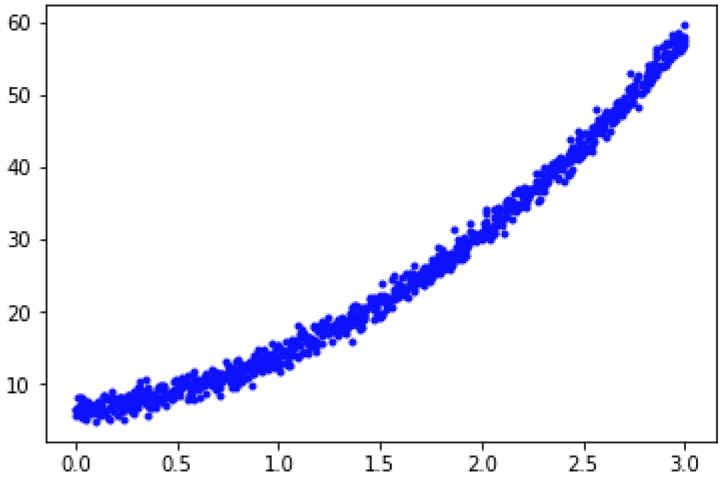
1. Comparative Analysis of Regression Models on Synthetic Data

a) Generation of dataset from a quadratic function with noise that includes 1000 samples

```
numSamples = 1000
noise = numpy.random.randn(numSamples, 1)
x = 3*numpy.random.rand(numSamples, 1)
y = 5*(x**2) + 2*x + 7 + noise

%matplotlib inline
plt.plot(x,y, "b.")
plt.show
```

```
<function matplotlib.pyplot.show>
```



b) Creation of a 80/20 train/set split of the dataset

```
# 1b
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=42)
print("Number samples in training: ", len(x_train))
print("Number samples in testing: ", len(x_test))
```

```
Number samples in training:  800
Number samples in testing:  200
```

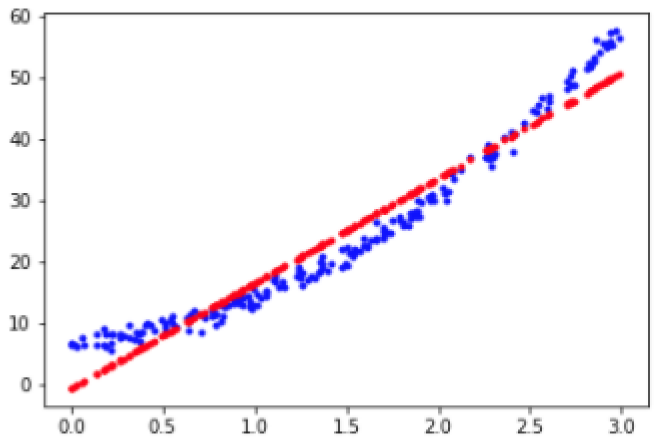
c) Training of the four regression models

i) Linear Model

```
lr_model = linear_model.LinearRegression().fit(x_train, y_train)
y_predicted = lr_model.intercept_ + lr_model.coef_*x_test

plt.plot(x_test,y_test, "b.")
plt.plot(x_test, y_predicted, "r.")
plt.show
```

```
<function matplotlib.pyplot.show>
```

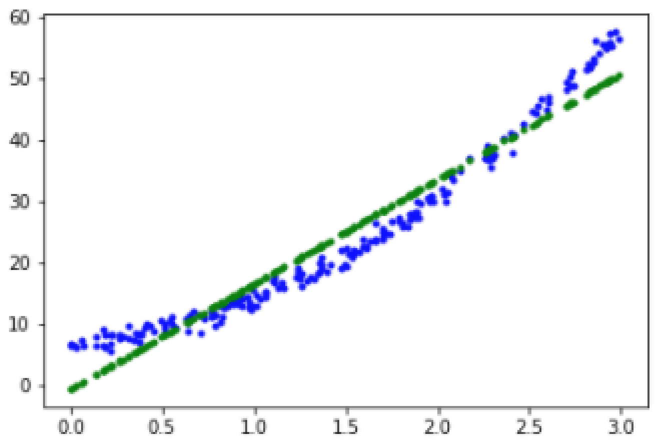


ii) Ridge Model (alpha = 0.5)

```
ridge_model = linear_model.Ridge(alpha = 0.5)
ridge_model.fit(x_train, y_train)
ridge_predicted = ridge_model.predict(x_test)

plt.plot(x_test, y_test, "b.")
plt.plot(x_test, ridge_predicted, "g.")
```

```
[<matplotlib.lines.Line2D at 0x7f283cd9fa20>]
```

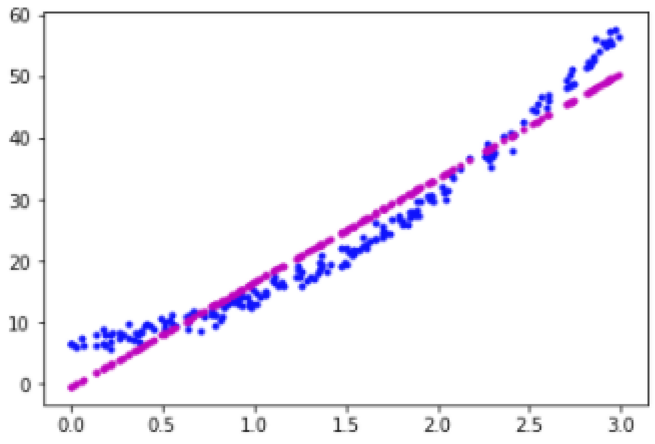


iii) Lasso Model (alpha = 0.1)

```
lasso_model = linear_model.Lasso(alpha = 0.1)
lasso_model. fit(x_train, y_train)
lasso_predicted = lasso_model.predict(x_test)

plt.plot(x_test, y_test, "b.")
plt.plot(x_test, lasso_predicted, "m.")
```

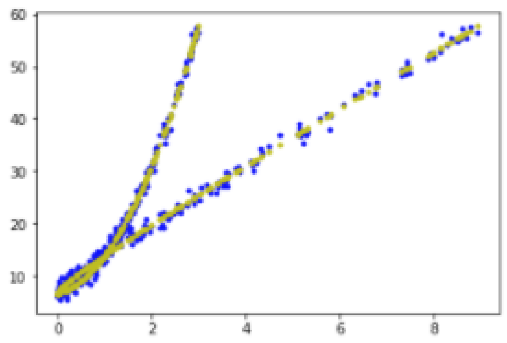
```
[<matplotlib.lines.Line2D at 0x7f283cccb278>]
```



iv) Polynomial Model (2nd degree)

```
poly_features = PolynomialFeatures(degree=2, include_bias = False)
x_poly = poly_features.fit_transform(x)

x_poly_train, x_poly_test, y_poly_train, y_poly_test = train_test_split(x_poly, y, test_size=0.2, random_state=42)
poly_model = linear_model.LinearRegression().fit(x_poly_train, y_poly_train)
y_poly_predicted = poly_model.predict(x_poly_test)
plt.plot(x_poly_test, y_poly_test, "b.")
plt.plot(x_poly_test, y_poly_predicted, "y.")
plt.show()
```



d) Predictive performance for each regression model

	Coefficient of determination	Coefficient of correlation	Mean Absolute Error
Linear Model	0.938905309091	0.96916606	3.10919118502
Ridge Model (alpha = 0.5)	0.938933851285	0.96916606	3.10761368229
Lasso Model (alpha = 0.1)	0.939117771233	0.969166060143	3.09434064844
Polynomial Model (2nd degree)	0.994824252391	0.99741968	0.841479971287

e) Analysis and Comparison of the models

The polynomial model has the lowest Mean Absolute Error out of the four models which means the predicted values of the model are closest to the true values of the dataset. The other three models have MAEs that are significantly larger but quite close to each other – with the lasso model performing marginally better than the linear and ridge algorithms. The polynomial model also has the largest Coefficient of correlation – extremely close to 1. All four models displayed a high positive correlation as all the values were above 0.8. Once again, the lasso model seemed to be marginally better than the linear and ridge.

Because the MAE is an average of the errors, it can be inferred that the polynomial model had lower bias than the other three models. All the models can be estimated to have a low variance since the correlation was high but the polynomial had a slightly higher value than the rest. Therefore, I think that the linear, ridge, and lasso models can be approximated to being under fitted because they had a higher bias and lower variance. The polynomial was a little more fitted than the rest (although still underfitted in general) because it had a higher variance and a lower bias. Another way to think about this would be to compare the MAEs of the training data with these MAEs values. When running the calculation (around 18 – see code for exact values), we find that even those MAEs are really high and therefore since both test and training sets have large mean absolute errors, it can be concluded that all the models are underfitted.

The method that performed the best was the polynomial model. It had the lowest mean absolute error as well as the highest correlation coefficient. This actually makes sense to me because we generated the function using a quadratic equation and therefore a second order polynomial should be able to best match the dataset.

2. Comparative Analysis of Regression Models on Real Data

a) Load real dataset

```
boston_data = load_boston()
```

b) Creation of a 80/20 train/set split of the dataset

```
x_train, x_test, y_train, y_test = train_test_split(boston_data.data, boston_data.target, test_size=0.2, random_state=42)

print("Number of samples in training set: ", len(x_train))
print("Number of samples in testing set: ", len(x_test))
```

```
Number of samples in training set:  404
Number of samples in testing set:  102
```

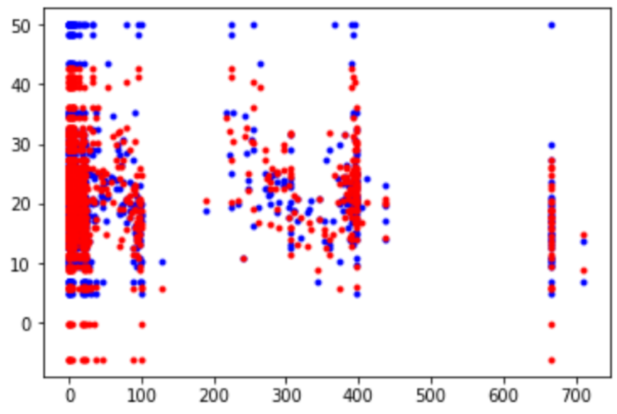
c) Training of the four regression models

i) Linear Model

```
lr_model = linear_model.LinearRegression().fit(x_train, y_train)
y_predicted = lr_model.predict(x_test)

plt.plot(x_test,y_test, "b.")
plt.plot(x_test, y_predicted, "r.")
plt.show
```

```
<function matplotlib.pyplot.show>
```

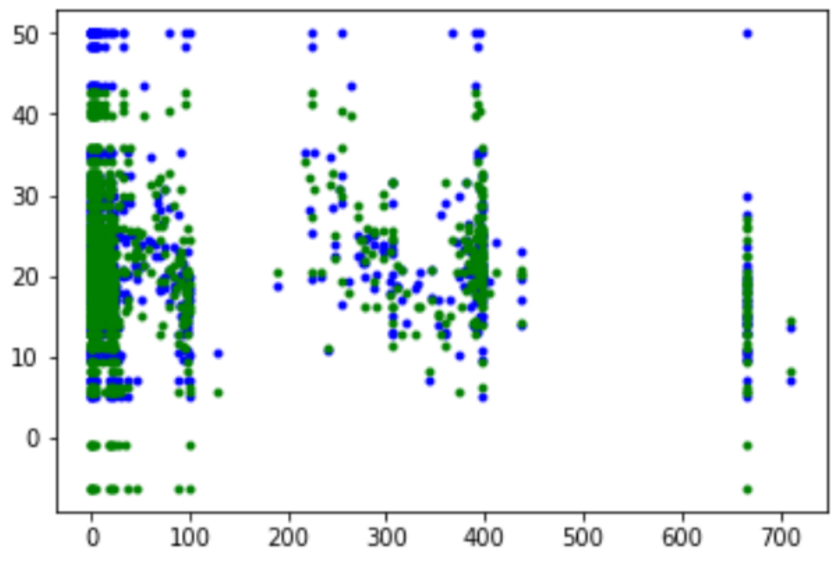


ii) Ridge Model (alpha = 0.5)

```
ridge_model = linear_model.Ridge(alpha = 0.5)
ridge_model.fit(x_train, y_train)
ridge_predicted = ridge_model.predict(x_test)

plt.plot(x_test, y_test, "b.")
plt.plot(x_test, ridge_predicted, "g.")
```

```
[<matplotlib.lines.Line2D at 0x7f795d7a91d0>,
 <matplotlib.lines.Line2D at 0x7f795d79c898>,
 <matplotlib.lines.Line2D at 0x7f795d79c208>,
 <matplotlib.lines.Line2D at 0x7f795d79ce80>,
 <matplotlib.lines.Line2D at 0x7f795d79cf98>,
 <matplotlib.lines.Line2D at 0x7f795d79ce48>,
 <matplotlib.lines.Line2D at 0x7f795d79ccf8>,
 <matplotlib.lines.Line2D at 0x7f795d79c0b8>,
 <matplotlib.lines.Line2D at 0x7f795d79c390>,
 <matplotlib.lines.Line2D at 0x7f795d7c75f8>,
 <matplotlib.lines.Line2D at 0x7f795d7c77b8>,
 <matplotlib.lines.Line2D at 0x7f795d7c77f0>,
 <matplotlib.lines.Line2D at 0x7f795d7c76d8>]
```

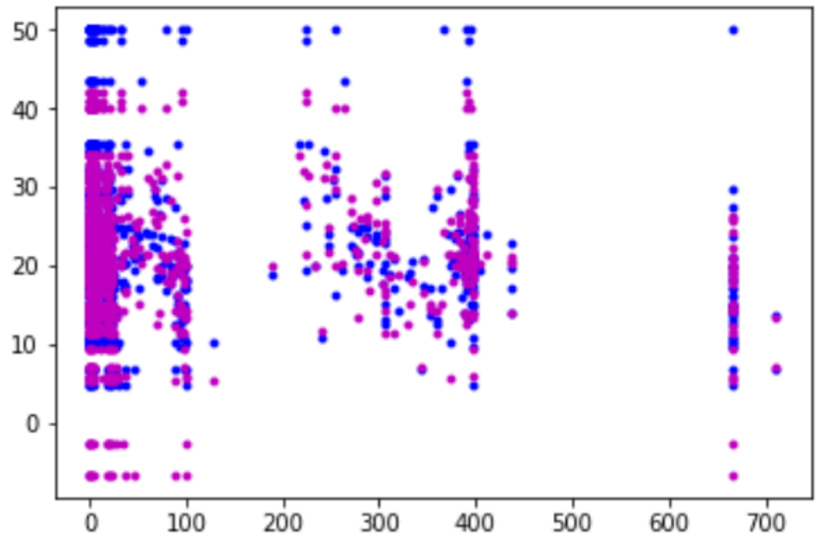


iii) Lasso Model (alpha = 0.1)

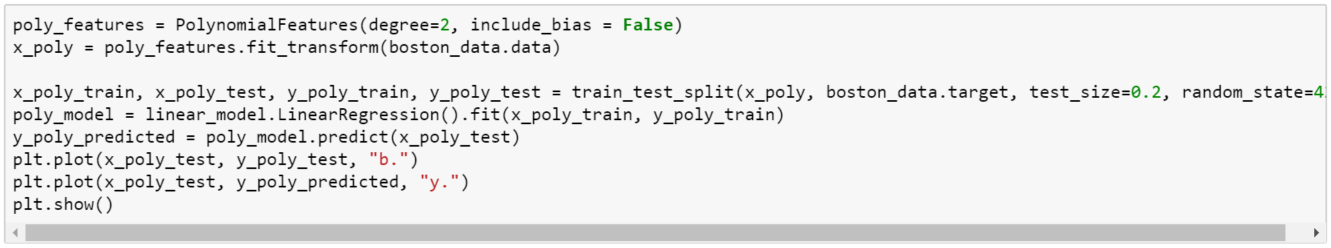
```
lasso_model = linear_model.Lasso(alpha = 0.1)
lasso_model. fit(x_train, y_train)
lasso_predicted = lasso_model.predict(x_test)

plt.plot(x_test, y_test, "b.")
plt.plot(x_test, lasso_predicted, "m.")
```

```
[<matplotlib.lines.Line2D at 0x7f795d7b02e8>,
 <matplotlib.lines.Line2D at 0x7f795e967278>,
 <matplotlib.lines.Line2D at 0x7f795e967828>,
 <matplotlib.lines.Line2D at 0x7f795e9675f8>,
 <matplotlib.lines.Line2D at 0x7f795e8ecf60>,
 <matplotlib.lines.Line2D at 0x7f795e8ec400>,
 <matplotlib.lines.Line2D at 0x7f795e8ec7f0>,
 <matplotlib.lines.Line2D at 0x7f795e8ecbe0>,
 <matplotlib.lines.Line2D at 0x7f795e8ecfd0>,
 <matplotlib.lines.Line2D at 0x7f795e8ec198>,
 <matplotlib.lines.Line2D at 0x7f795e8ec390>,
 <matplotlib.lines.Line2D at 0x7f795e8ec588>,
 <matplotlib.lines.Line2D at 0x7f795e8ec780>]
```

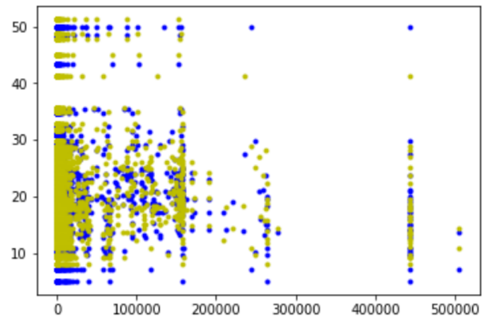


iv) Polynomial Model (2nd degree)



```
poly_features = PolynomialFeatures(degree=2, include_bias = False)
x_poly = poly_features.fit_transform(boston_data.data)

x_poly_train, x_poly_test, y_poly_train, y_poly_test = train_test_split(x_poly, boston_data.target, test_size=0.2, random_state=4
poly_model = linear_model.LinearRegression().fit(x_poly_train, y_poly_train)
y_poly_predicted = poly_model.predict(x_poly_test)
plt.plot(x_poly_test, y_poly_test, "b.")
plt.plot(x_poly_test, y_poly_predicted, "y.")
plt.show()
```



d) Predictive performance for each regression model

	Coefficient of determination	Coefficient of correlation	Mean Absolute Error
Linear Model	0.668482575397	0.82736437122	3.19150897227
Ridge Model (alpha = 0.5)	0.667316025799	0.82706692383	3.15156813384
Lasso Model (alpha = 0.1)	0.656705559418	0.821970953171	3.14633829462
Polynomial Model (2nd degree)	0.805959530563	0.900285122095	2.57537984978

e) Analysis and Comparison of the models

With the real dataset, the Mean Absolute Errors were closer to each other although once again, the polynomial model has the lowest errors in comparison to the true values. This time though, the linear model performed worst in terms of the error – with the lasso and ridge quite close to each other. The polynomial model also had a better correlation coefficient than the other models, but not nearly has strong as the simulated data. All four models can still be considered to be strongly positively correlated. The lasso model actually performed the worst in this measure.

As stated earlier, the mean absolute error is an indicator of bias and again, the polynomial model seems to have less bias than the others and a higher variance. Therefore, I would think that the linear, ridge, and lasso models are slightly under fit. The polynomial model had the lowest MAE, but when compared to the calculated mean absolute error of the training set (around 10 – see code for exact values) it can be seen that all four models are underfit as both MAEs are high.

I think the polynomial model performed the best again because it had the lowest mean absolute error as well as the highest correlation coefficient meaning it most accurately represented the true values. That being said, the MAE of the polynomial was higher than the MAE of the polynomial model of the simulated data. Although, the polynomial model performs the best out of the four models built, I think it is still slightly underfitted and fails do provide enough complexity to model the dataset. A polynomial model of a higher order would perform better in my opinion.

3. Hyper parameters of Regularized Regression Models

a) Ridge Regression - Impact of learned model when alpha = 0

As alpha goes to 0 in a ridge regression, the higher the magnitude of the coefficients. Thus the model would be not regularized at all and would simply become an Ordinary Least Square Regression model. The learned models would thus become highly over fitted and the algorithm would learn all the noise from the training set.

b) Ridge Regression - Impact of learned model when alpha = 1

As alpha goes to 1 in a ridge regression, the magnitude of the coefficients disappears altogether. Therefore, in this scenario, the model would be too regularized and become a somewhat linear model. The smoothness constraint would be very high because the flexibility of fit would be very strict. The learned model would be under fitted and the algorithm would not learn enough from the training set.

c) Lasso Regression - Impact of learned model when alpha = 0

As alpha goes to 0 in a lasso regression, the magnitudes of the coefficient become the same as a linear regression. This will mean we are not regularizing our model to penalize insignificant features. The learned model will take into account all the features even ones that do not have a significant impact and therefore the benefit of a lasso regression is lost.

d) Lasso Regression - Impact of learned model when alpha = 1

As alpha goes to 1 in a lasso regression, more and more feature coefficients go to 0 and do not play a part in the algorithm. This will mean we are regularizing the mode too much and penalizing all the features. The learned model will not take into account any of the features and eventually the model would become a flat plane – not being affected by any of the features.