Name: Karan Dua
Student Id: 21331391

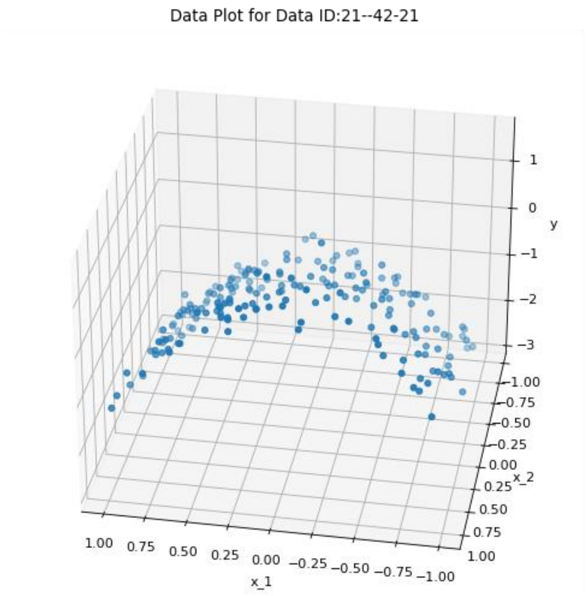
Course Code: CS7CS4-202223 MACHINE LEARNING

Question (i)(a) Reading the data and visualising the data on a 3-d plot

To read the data, I have used pandas library in python. The dataset had a comment, so I have removed it using the inbuilt feature of this function. Please refer below head of the dataset**. My data id is: 21--42-21**:

	x1	x2	y
0	-0.45	-0.72	-1.025864
1	0.11	0.21	0.056347
2	-0.89	-0.68	-1.889089
3	0.84	-0.12	-1.427458
4	-0.42	-0.37	-0.187069

I have used scatterplot to plot a 3-d projection of the data with x1 on the x-axis, x2 on the y-axis and y on the z-axis. Please refer below the 3-D projection.



By analysing the plot, I can clearly see that the training data lies on curve. This implies that relationship between dependent variable y and independent variables x1 and x2 is non-linear.

Question (i)(b) Adding Polynomial Features to the data, training Lasso Regression, reporting parameters and result discussion

To add polynomial features up to power 5 for x1 and x2, I have used PolynomialFeature function of sklearn library. Now, I have used this new data set to train the Lasso Regression model. Lasso Regression model is a Linear Regression model which is regularised as it contains an L1 penalty parameter. In our case, I have taken the following penalty parameters: [1, 5, 10, 100, 500, 1000]. I have check for C values less than 1 also, but for these all the coefficients were 0 therefore, I have removed them from final result. Using these C values, I have trained the Lasso regression model and extracted the following model parameters:

	Penalty	Intercept	1	x1	x2	x1^2	x1 x2	x2^2	x1^3	x1^2 x2	x1 x2^2	x2^3	x1^4	x1^3 x2	x1^2 x2^2	x1 x2^3	x2^4	x1^5	x1^4 x2	x1^3 x2^2	x1^2 x2^3	x1 x2^4	x2^5
0	1	[-0.697]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[0.0]	[-0.0]	[0.0]	[0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[0.0]	[0.0]
1	5	[-0.401]	[0.0]	[-0.0]	[0.741]	[-0.825]	[0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]
2	10	[-0.194]	[0.0]	[-0.0]	[0.871]	[-1.426]	[-0.0]	[-0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]	[-0.0]	[0.0]
3	100	[-0.004]	[0.0]	[-0.036]	[0.991]	[-1.965]	[-0.025]	[-0.008]	[-0.0]	[-0.0]	[-0.0]	[0.0]	[-0.0]	[-0.0]	[-0.006]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[-0.0]	[0.0]
4	500	[0.015]	[0.0]	[-0.049]	[1.048]	[-1.965]	[-0.057]	[-0.0]	[0.0]	[-0.145]	[-0.0]	[0.012]	[-0.023]	[-0.0]	[-0.088]	[-0.0]	[-0.03]	[0.0]	[-0.0]	[-0.0]	[0.0]	[0.0]	[0.0]
5	1000	[0.015]	[0.0]	[-0.07]	[1.065]	[-1.943]	[-0.048]	[-0.0]	[0.039]	[-0.209]	[-0.01]	[0.0]	[-0.052]	[0.0]	[-0.107]	[-0.02]	[-0.029]	[-0.0]	[-0.016]	[-0.0]	[0.081]	[0.0]	[-0.0]

Explanation of above result with regards to Lasso Model equation:
Our model is polynomial of degree 5. Therefore, its equation can be written as:
m0 * 1 + m1*x1 + m2*x2 + m3*x1^2 + m4*x1x2 + m5*x2^2 + ……. + m20*x1x2^4 + m21*x2^5 + intercept, where m1, m2, m3 are our model coefficients.
Now, for C=1, our model has all 0 coefficients, so model equation will look like 0*1 + 0*x1 + 0*x2 + 0*x1^2 + 0*x1x2 + 0*x2^2 + ……. + 0*x1x2^4 + 0*x2^5 -0.697 = -0.697 which is a constant.

For C=5, our model equation will look like, 0*1 + 0*x1 + 0.741*x2 - 0.825*$x1^2$ – 0*x1x2 + 0*$x2^2$ + ……. + 0*$x1x2^4$ + 0*$x2^5$ -0.401 = 0.741*x2 - 0.825*$x1^2$ - 0.401 which is a quadratic equation of $x1^2$
Similarly, I have calculated model equation for all penalty parameters:

	Penalty	Model Equation
0	1	(-0.697)
1	5	(0.741) * x2 + (-0.825) * x1^2 + (-0.401)
2	10	(0.871) * x2 + (-1.426) * x1^2 + (-0.194)
3	100	(-0.036) * x1 + (0.991) * x2 + (-1.965) * x1^2 + (-0.025) * x1 x2 + (-0.008) * x2^2 + (-0.006) * x1^2 x2^2 + (-0.004)
4	500	(-0.049) * x1 + (1.048) * x2 + (-1.965) * x1^2 + (-0.057) * x1 x2 + (-0.145) * x1^2 x2 + (0.012) * x2^3 + (-0.023) * x1^4 + (-0.088) * x1^2 x2^2 + (-0.03) * x2^4 + (0.015)
5	1000	(-0.07) * x1 + (1.065) * x2 + (-1.943) * x1^2 + (-0.048) * x1 x2 + (0.039) * x1^3 + (-0.209) * x1^2 x2 + (-0.01) * x1 x2^2 + (-0.052) * x1^4 + (-0.107) * x1^2 x2^2 + (-0.02) * x1 x2^3 + (-0.029) * x2^4 + (-0.016) * x1^4 x2 + (0.081) * x1^2 x2^3 + (0.015)

From the above table, I can clearly analyse that penalty parameter has a significant impact on the model's coefficients. The cost function of Lasso Regression with L1 penalty parameter is defined as:

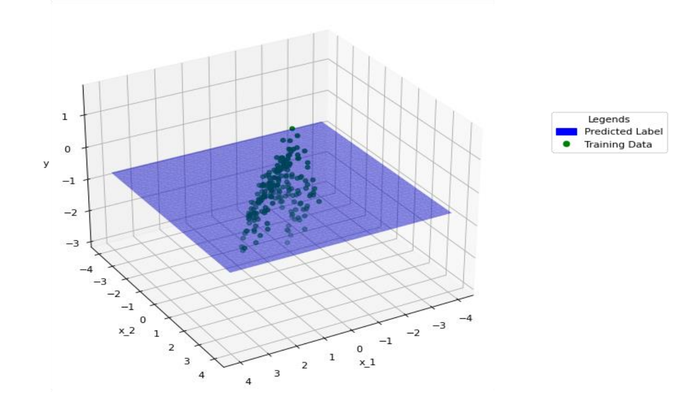
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \theta^T\theta/C$$

Based on this equation, we can clearly see that decreasing the value of penalty parameter C increases the value of Penalty in cost function, which leads to shrinking of the model coefficients. As we increase the value of C, less importance is given to penalty in cost function and therefore we have observed more coefficients in the model. For example, for C=1, penalty parameter had the highest influence in overall model, therefore all the model coefficients were reduced to 0 and we got model equation as y = -0.697. As value of C increased from 1 to 5, we see more non zero coefficients in model confirming that model is giving more significance to features in the train data and we get model equation as: y = 0.741*$x1^2$ - 0.825*$x1^2$ - 0.401. Furthermore, as value of C increases to 500, we get 9 non zero coefficients in model and for C=1000, we get 13 non zero coefficients confirming that more significance is being given to features in the data as value of C increases

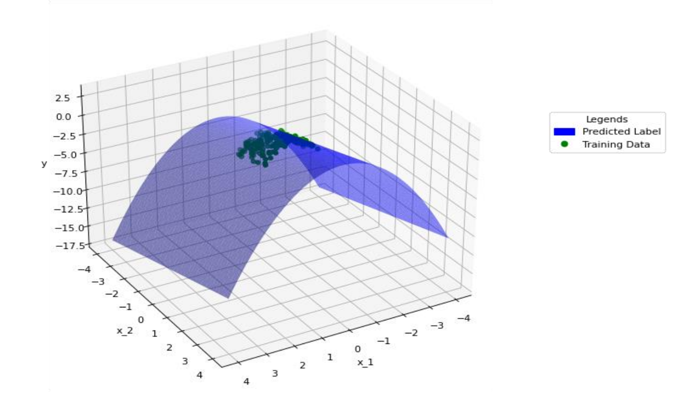
Question (i)(c) Plot of predictions and result discussion

To generate test features, I have first checked range of features in my data set. The range comes from -1 to 1, therefore I have generated test features in the range of -4 to 4. Then I used my trained model to predict values for the test features. Following are the plots generated for this task:

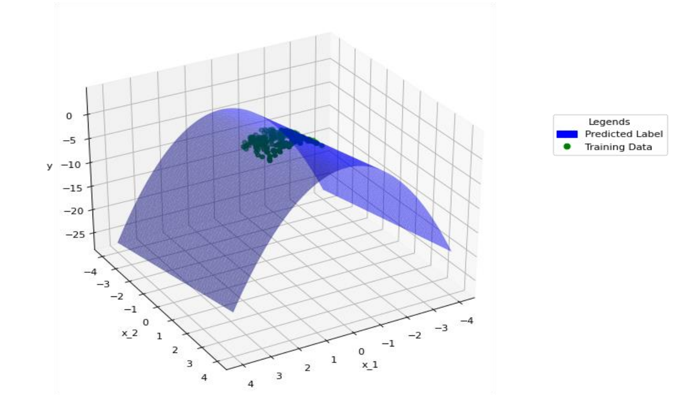
Data Plot for Actual VS Predicted value of y for Lasso Regression with Penalty Score: 1



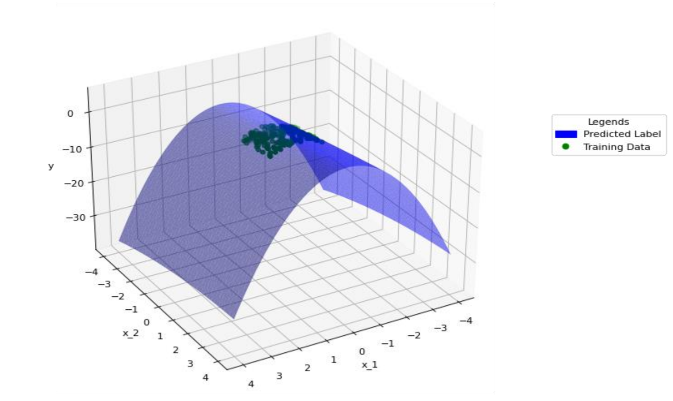
Data Plot for Actual VS Predicted value of y for Lasso Regression with Penalty Score: 5

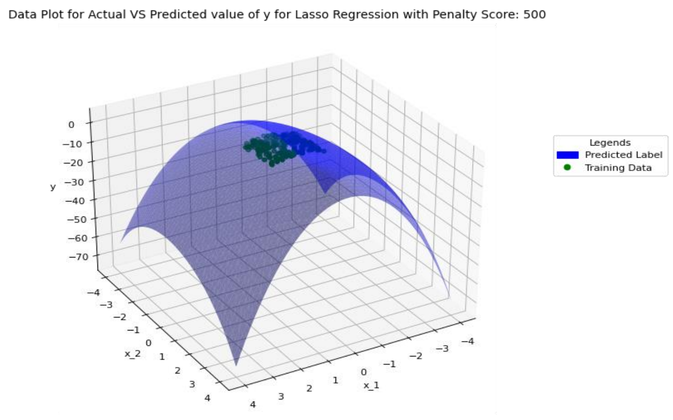


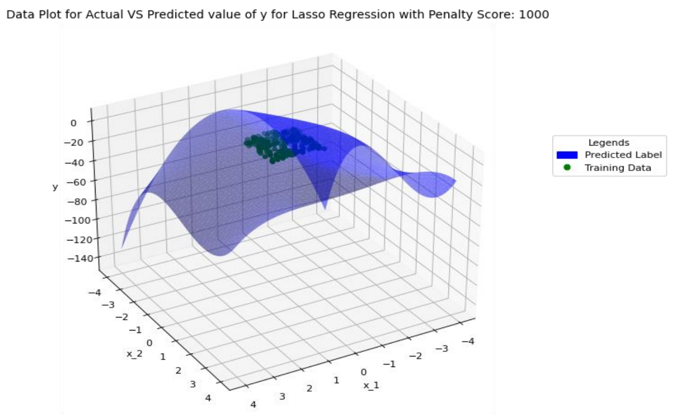
Data Plot for Actual VS Predicted value of y for Lasso Regression with Penalty Score: 10



Data Plot for Actual VS Predicted value of y for Lasso Regression with Penalty Score: 100







We can analyse from the plots that for C=1, our model coefficients were all 0 therefore model predicts a constant value of intercept (-0.697) for all test data. As value of C increase to 5, 10, 100 the model puts more significance to features in the model rather than penalty, we see a better fit for the predictions. This means that model is able to accommodate for the non-linearity in the data as we see higher degree polynomials included in model. But as value of C increases further to C=500, 1000 we see model getting more and more complex as more higher degree polynomials are added and therefore, we see very complex surfaces for predictions in the plots.

Question (i)(d) Overfitting and Underfitting and use of C for its trade-off

Underfitting refer to situation when a trained model failed to accommodate underlying complexity in the training data. Such models are too simple as they fail to analyse the important features in the dataset provided for training and may ignore these features in the final model equation. The performance of these models is very poor on both test and training data.

Overfitting refers to a situation when a model tries to fit too much to training data. In doing so, model tries to fit itself for the outliers and noise present in the data too. Such models often tens to be too complex as they try to fit themselves for all the points in training data rather than the ones that are significant. These models perform extremely well on training data as it has accommodated all the variations while training but fails to perform on test data due to noise it has captured during the training process.

In our model also we can clearly see the impact that penalty parameter C had on the overall model performance. For very small value of C, model has neglected all features in the training data and that has clearly resulted in underfitting as model always predicted a constant value for all test data. The reason is that model has failed to capture underlying relationship between the features in the model equation. However, for large values of C, like 500, 1000, the model has tried to overfit the training data as visible from the complex plane of the graph, it clearly shows that model is too complex in trying to over-compensate the features in training data. It has 9 and 13 non zero coefficients respectively. This means that many high degree polynomials are now part of the model equation making it too complex highlighting that model is trying to use all the features even when weak or no relationship exist between these features. The main reason for this could be that model is trying to accommodate noise into the model, hence it is clearly overfitted. For C=5, 10, we get relatively simple polynomial equations an in plots generated we can see that surface is relatively simple and adequately captures the non-linear relationship between features and label. For simple model, we should always choose smaller value of C. Therefore, C=5 seems ideal choice to achieve the trade-off between overfitting and underfitting.

Question (i)(e) Ridge Regression and Comparison with Lasso Regression

I have used the same process to train the Ridge Regression model as Lasso regression. Same training data was used as Lasso Regression and Penalty Parameters used during training were: [1, 5, 10, 100, 500, 1000]. Following are the parameters extracted from the model after training.

Name: Karan Dua
Student Id: 21331391

Course Code: CS7CS4-202223 MACHINE LEARNING

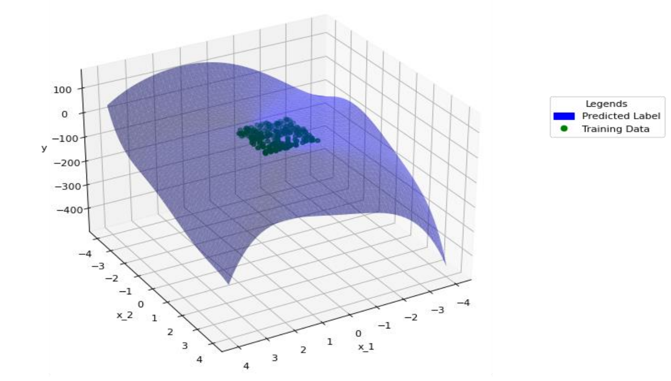
	Penalty	Intercept	1	x1	x2	x1^2	x1 x2	x2^2	x1^3	x1^2 x2	x1 x2^2	x2^3	x1^4	x1^3 x2	x1^2 x2^2	x1 x2^3	x2^4	x1^5	x1^4 x2	x1^3 x2^2	x1^2 x2^3	x1 x2^4	x2^5
0	1	[-0.053]	[0.0]	[-0.092]	[0.98]	[-1.512]	[-0.058]	[0.044]	[0.118]	[-0.113]	[-0.116]	[0.207]	[-0.461]	[0.05]	[-0.237]	[-0.042]	[-0.02]	[-0.04]	[-0.148]	[0.017]	[0.112]	[0.108]	[-0.133]
1	5	[-0.007]	[0.0]	[-0.139]	[1.053]	[-1.824]	[-0.082]	[0.072]	[0.391]	[-0.32]	[-0.258]	[0.213]	[-0.157]	[0.075]	[-0.178]	[-0.048]	[-0.087]	[-0.286]	[-0.141]	[-0.039]	[0.368]	[0.318]	[-0.251]
2	10	[0.002]	[0.0]	[-0.169]	[1.072]	[-1.902]	[-0.087]	[0.088]	[0.546]	[-0.406]	[-0.305]	[0.224]	[-0.073]	[0.075]	[-0.167]	[-0.045]	[-0.113]	[-0.422]	[-0.112]	[-0.077]	[0.453]	[0.399]	[-0.292]
3	100	[0.012]	[0.0]	[-0.23]	[1.096]	[-1.999]	[-0.095]	[0.115]	[0.831]	[-0.54]	[-0.358]	[0.25]	[0.034]	[0.069]	[-0.155]	[-0.036]	[-0.152]	[-0.667]	[-0.048]	[-0.147]	[0.569]	[0.51]	[-0.355]
4	500	[0.013]	[0.0]	[-0.238]	[1.098]	[-2.009]	[-0.095]	[0.118]	[0.87]	[-0.557]	[-0.363]	[0.254]	[0.046]	[0.069]	[-0.154]	[-0.034]	[-0.157]	[-0.701]	[-0.039]	[-0.157]	[0.582]	[0.523]	[-0.364]
5	1000	[0.013]	[0.0]	[-0.239]	[1.099]	[-2.011]	[-0.096]	[0.119]	[0.875]	[-0.559]	[-0.364]	[0.255]	[0.047]	[0.068]	[-0.153]	[-0.034]	[-0.158]	[-0.705]	[-0.037]	[-0.158]	[0.584]	[0.525]	[-0.365]

Following are the model equations generated based on above parameters:

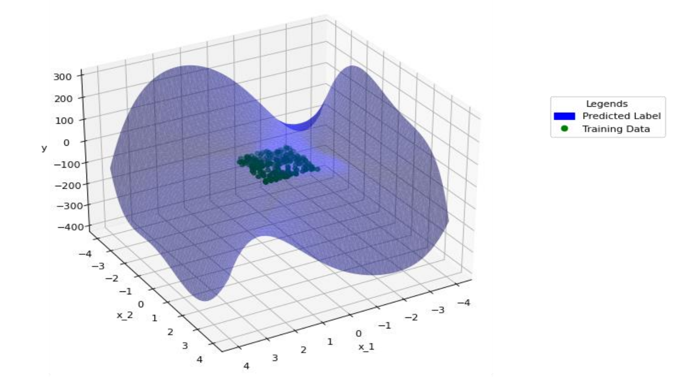
	Penalty	Model Equation
0	1	(-0.092) * x1 + (0.98) * x2 + (-1.512) * x1^2 + (-0.058) * x1 x2 + (0.044) * x2^2 + (0.118) * x1^3 + (-0.113) * x1^2 x2 + (-0.116) * x1 x2^2 + (0.207) * x2^3 + (-0.461) * x1^4 + (0.05) * x1^3 x2 + (-0.237) * x1^2 x2^2 + (-0.042) * x1 x2^3 + (-0.02) * x2^4 + (-0.04) * x1^5 + (-0.148) * x1^4 x2 + (0.017) * x1^3 x2^2 + (0.112) * x1^2 x2^3 + (0.108) * x1 x2^4 + (-0.133) * x2^5 + (-0.053)
1	5	(-0.139) * x1 + (1.053) * x2 + (-1.824) * x1^2 + (-0.082) * x1 x2 + (0.072) * x2^2 + (0.391) * x1^3 + (-0.32) * x1^2 x2 + (-0.258) * x1 x2^2 + (0.213) * x2^3 + (-0.157) * x1^4 + (0.075) * x1^3 x2 + (-0.178) * x1^2 x2^2 + (-0.048) * x1 x2^3 + (-0.087) * x2^4 + (-0.286) * x1^5 + (-0.141) * x1^4 x2 + (-0.039) * x1^3 x2^2 + (0.368) * x1^2 x2^3 + (0.318) * x1 x2^4 + (-0.251) * x2^5 + (-0.007)
2	10	(-0.169) * x1 + (1.072) * x2 + (-1.902) * x1^2 + (-0.087) * x1 x2 + (0.088) * x2^2 + (0.546) * x1^3 + (-0.406) * x1^2 x2 + (-0.305) * x1 x2^2 + (0.224) * x2^3 + (-0.073) * x1^4 + (0.075) * x1^3 x2 + (-0.167) * x1^2 x2^2 + (-0.045) * x1 x2^3 + (-0.113) * x2^4 + (-0.422) * x1^5 + (-0.112) * x1^4 x2 + (-0.077) * x1^3 x2^2 + (0.453) * x1^2 x2^3 + (0.399) * x1 x2^4 + (-0.292) * x2^5 + (0.002)
3	100	(-0.23) * x1 + (1.096) * x2 + (-1.999) * x1^2 + (-0.095) * x1 x2 + (0.115) * x2^2 + (0.831) * x1^3 + (-0.54) * x1^2 x2 + (-0.358) * x1 x2^2 + (0.25) * x2^3 + (0.034) * x1^4 + (0.069) * x1^3 x2 + (-0.155) * x1^2 x2^2 + (-0.036) * x1 x2^3 + (-0.152) * x2^4 + (-0.667) * x1^5 + (-0.048) * x1^4 x2 + (-0.147) * x1^3 x2^2 + (0.569) * x1^2 x2^3 + (0.51) * x1 x2^4 + (-0.355) * x2^5 + (0.012)
4	500	(-0.238) * x1 + (1.098) * x2 + (-2.009) * x1^2 + (-0.095) * x1 x2 + (0.118) * x2^2 + (0.87) * x1^3 + (-0.557) * x1^2 x2 + (-0.363) * x1 x2^2 + (0.254) * x2^3 + (0.046) * x1^4 + (0.069) * x1^3 x2 + (-0.154) * x1^2 x2^2 + (-0.034) * x1 x2^3 + (-0.157) * x2^4 + (-0.701) * x1^5 + (-0.039) * x1^4 x2 + (-0.157) * x1^3 x2^2 + (0.582) * x1^2 x2^3 + (0.523) * x1 x2^4 + (-0.364) * x2^5 + (0.013)
5	1000	(-0.239) * x1 + (1.099) * x2 + (-2.011) * x1^2 + (-0.096) * x1 x2 + (0.119) * x2^2 + (0.875) * x1^3 + (-0.559) * x1^2 x2 + (-0.364) * x1 x2^2 + (0.255) * x2^3 + (0.047) * x1^4 + (0.068) * x1^3 x2 + (-0.153) * x1^2 x2^2 + (-0.034) * x1 x2^3 + (-0.158) * x2^4 + (-0.705) * x1^5 + (-0.037) * x1^4 x2 + (-0.158) * x1^3 x2^2 + (0.584) * x1^2 x2^3 + (0.525) * x1 x2^4 + (-0.365) * x2^5 + (0.013)

It can be clearly analysed that even for small value of C=1, the model generated is fairly complex and number of non-zero coefficients does not change for C=1 to C=1000. This means that there are many high degree polynomials present in the model equation. The role of Penalty parameter can be seen in absolute magnitude of coefficients as increase in C has also increased the absolute magnitude of the coefficients. By looking at the coefficients it can be assumed that the scatterplot generated for test data should be fairly complex.

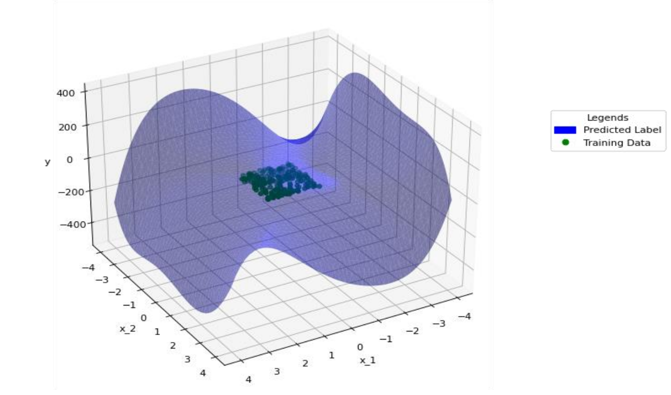
Data Plot for Actual VS Predicted value of y for Ridge Regression with Penalty Score: 1



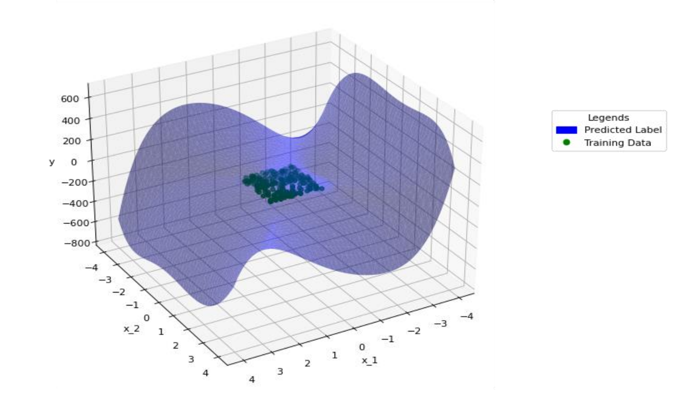
Data Plot for Actual VS Predicted value of y for Ridge Regression with Penalty Score: 5

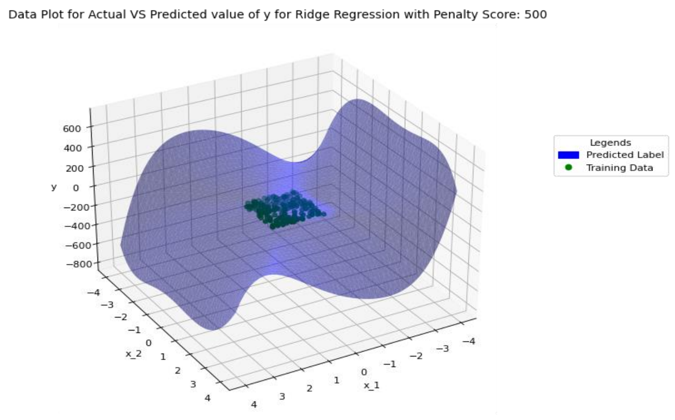


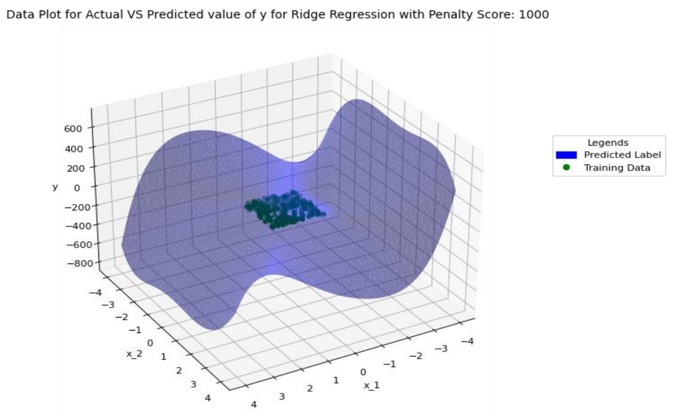
Data Plot for Actual VS Predicted value of y for Ridge Regression with Penalty Score: 10



Data Plot for Actual VS Predicted value of y for Ridge Regression with Penalty Score: 100





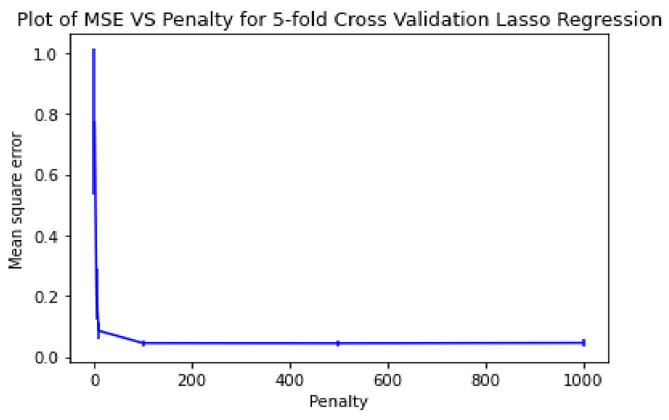


It can be clearly analysed from the plots that these are fairly complex surfaces as there are higher degree polynomials involved in the model equation but one this that is distinct is that As C increases from 1 to 1000 the surface becomes less smooth indicating the higher complexity of the model and also possibly overfitting by the model for noise in the data.

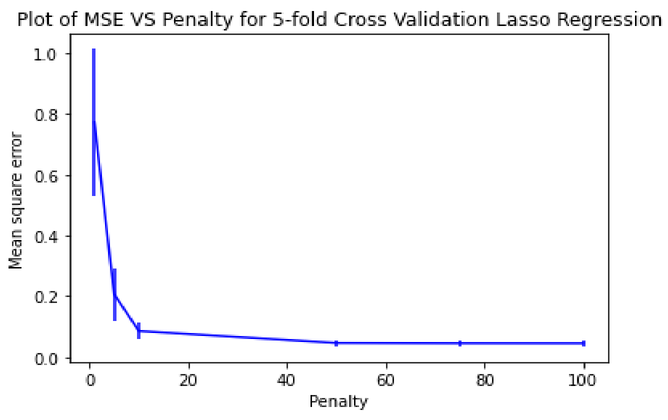
On comparing both the models, we can clearly say that for small values of Lasso Regression clearly removes all the less important features from the model, making it much simpler and avoids the overfitting problem. Ridge regression does not remove less important features even for smaller magnitude of penalty term; hence the final model equation is way more complex and less analysable compared to Lasso Regression. Thus, we can conclude that for feature selection process, we should use Lasso regression as it pays importance to highly related features in the model and completely reduces all other unrelated features to 0. The model generated is fairly simple and provides a good trade-off between underfitting and overfitting.

Question (ii)(a) 5-fold Cross Validation for Lasso Regression for different values of Penalty

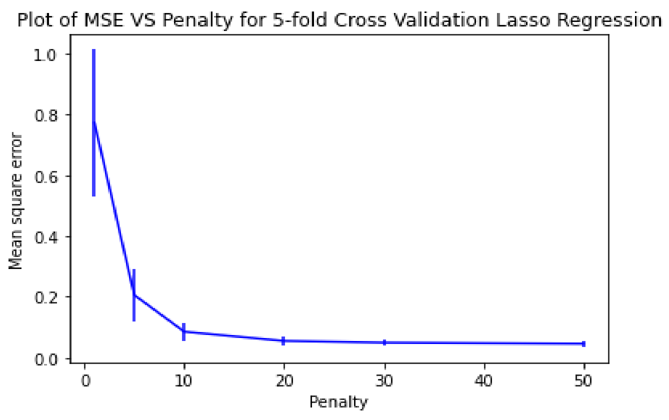
I have used multiple iterations to identify the correct value of Penalty parameter C for Lasso Regression model. In each iteration I have passed an array of penalty parameters. Then, for each penalty parameter in array, I have used 5-fold cross validation strategy to train Lasso regression model for given penalty parameter. Then, I have used this model to predict value using test features. Using predicted value, I have calculated Mean Square Error and Standard Deviation of the trained model. Finally, I have plotted a plot Mean Square Error and Penalty Parameter. Using matplotlib errorbar plot, I have also shown Standard Deviation of the model. Below are various plots that I have generated for different array of penalty parameters.



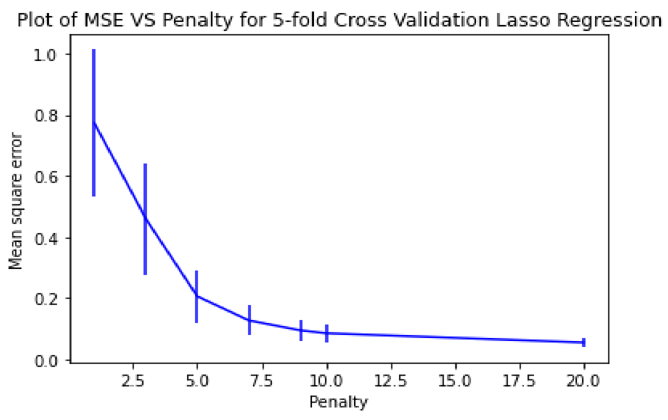
Plot 1: C = [1, 5, 10, 100, 500, 1000]



Plot 2: C = [1, 5, 10, 50, 75, 100]



Plot 3: C = [1, 5, 10, 20, 30, 50]



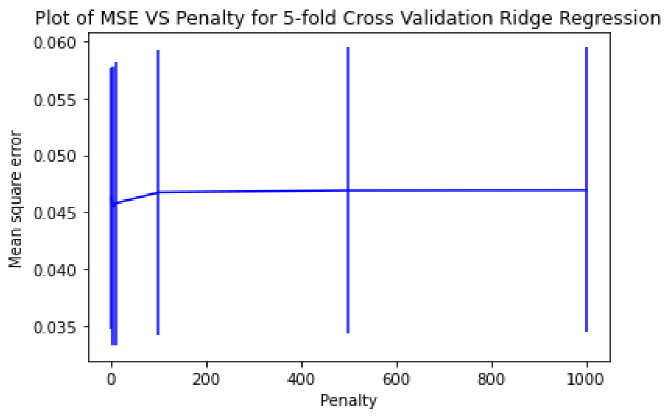
Plot 4: C = [1, 3, 5, 7, 9, 10, 20]

Question (ii)(b) Recommended value of Penalty Parameter based on above plots

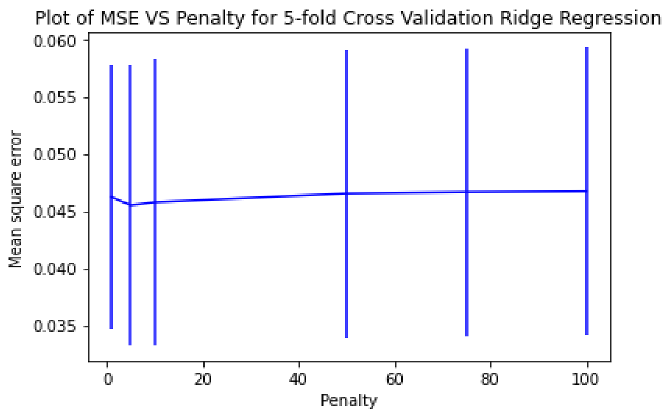
I have created Plot 1 on broad range of penalty parameters starting from 1 to 1000. By analysing this plot, I can see that Mean Square Error has converged by C=100. Again, I used same function to zoom into the plot by checking plot for penalty parameter 1 to 100. By analysing this plot, I can say that again MSE has converged by C=50. I have again plotted for C range 1 to 50 and then 1 to 20. Plot 4 shows the elbow like structure that I was looking for in the plot. 'Elbow Method' is used to identify a tipping pint in graph where the graph starts to converge. In simple terms, before 'Elbow' point, for small change in C, value of MSE increases/decreases sharply and after 'Elbow' point, even for large change in C, value of MSE does not change much. Therefore, it is an optimal point where the graph has just started to converge. Using 'Elbow Method' I can clearly spot that penalty parameter 5 is the optimal parameter for Lasso Regression. The primary reason for this is that MSE does not reduce significantly for C greater than 5 and for C less than 5 value of MSE is very high. Now, MSE plays a very important role in identifying the correct value of Penalty Parameter for regression model. Very low value of MSE, like in our case for C greater than 5, generally means that model is trying to overfit the data to reduce the error and very low value of MSE, like in our case for C less than 5, generally means that model is underfitting the data. Therefore, penalty parameter 5 seems to be the optimal value for balancing the trade-off between the underfitting of data and overfitting of data.

Question (ii)(c) 5-fold Cross Validation for Lasso Regression for different values of Penalty and Recommended value of Penalty Parameter

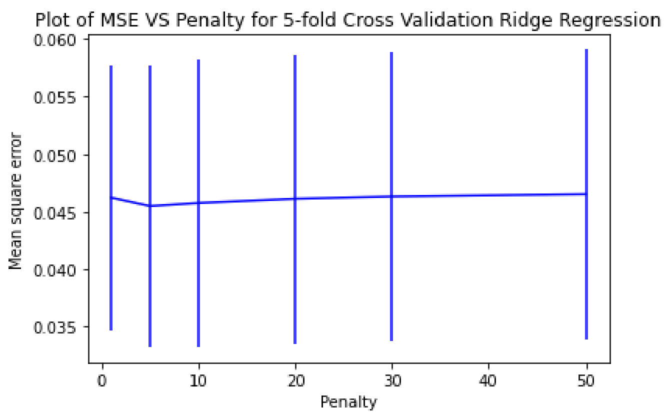
I have same process to generate graphs for wide range of Penalty parameter C and then reducing the range till I could spot clearly the elbow kind of structure in the graph. Below are the graphs generated for Ridge regression.



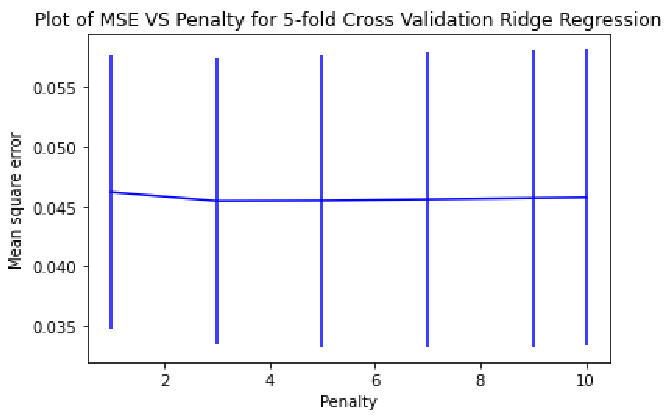
Plot 1: C = [1, 5, 10, 100, 500, 1000]



Plot 2: C = [1, 5, 10, 50, 75, 100]



Plot 3: C = [1, 5, 10, 20, 30, 50]



Plot 4: C = [1, 3, 5, 7, 9, 10]

For Ridge regression, the elbow like structure can be identified from Plot 4. For values greater than C=3, the graph has converged and for values less then C=3, the MSE increases. At, C=3, the value of MSE is at its lowest and Standard Deviation is also similar when compared to other points on the graph. Therefore, for ridge regression, the optimal value of penalty parameter is 3.

Name: Karan Dua Course Code: CS7CS4-202223 MACHINE LEARNING
Student Id: 21331391

Appendix: Code for Question 1 - All Parts

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plot
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from IPython.display import display
import math
from matplotlib.lines import Line2D
import matplotlib.patches as mpatches

df = pd.read_csv("data_week3.csv", names=["x1", "x2", "y"], header=None, comment='#')
display(df.head())
x1 = df.iloc[:,0]
x2 = df.iloc[:,1]
x = np.column_stack((x1, x2))
y = df.iloc[:,2]

figure = plot.figure(figsize=(8, 8), dpi = 80)
plot_final = figure.add_subplot(111, projection = '3d')
plot_final.scatter(x1, x2, y)
plot_final.set_xlabel("x_1")
plot_final.set_ylabel("x_2")
plot_final.set_zlabel("y")
plot_final.set_title('Data Plot for Data ID:21--42-21')
plot_final.view_init(azim = 100)

penalty_parameters = [1, 5, 10, 100, 500, 1000]

poly_features_function = PolynomialFeatures(5)
x_poly_features  = poly_features_function.fit_transform(x)

temp_df = pd.DataFrame(x, columns = ['x1', 'x2'])
feature_names = poly_features_function.get_feature_names(temp_df.columns)
feature_names.insert(0, 'Penalty')
feature_names.insert(1, 'Intercept')

lasso_model_dictionary = {}

lasso_model_params_df = pd.DataFrame(columns = feature_names)
lasso_model_equation_df = pd.DataFrame(columns = ['Penalty', 'Model Equation'])
for penalty in penalty_parameters:
    lasso_model = Lasso(alpha = 1 / (2 * penalty))
    lasso_model.fit(x_poly_features, y)

    model_dict = {}
    model_dict['Penalty'] = penalty
    model_dict['Intercept'] = [np.around(lasso_model.intercept_, decimals = 3)]
    for i in range(2,23) :
        model_dict[feature_names[i]] = [np.around(lasso_model.coef_[i-2], decimals = 3)]

    lasso_model_params_df = lasso_model_params_df.append(model_dict, ignore_index = True)

    model_eq_dict = {}
    model_eq_dict['Penalty'] = penalty
    equation_string = ''
    for i in range(2,23) :
        coeff = np.around(lasso_model.coef_[i-2], decimals = 3)
        if coeff != 0 :
            equation_string += '(' + str(coeff) + ')' + ' * ' + feature_names[i] + ' + '
    equation_string += '(' + str(np.around(lasso_model.intercept_, decimals = 3)) + ')'
    model_eq_dict['Model Equation'] = equation_string

    lasso_model_equation_df = lasso_model_equation_df.append(model_eq_dict, ignore_index = True)

    lasso_model_dictionary[penalty] = lasso_model

lasso_model_params_df = lasso_model_params_df.style.applymap(lambda x:'white-space:nowrap')
display(lasso_model_params_df)

lasso_model_equation_df = lasso_model_equation_df.style.set_properties(**{'text-align': 'left'})
lasso_model_equation_df = lasso_model_equation_df.applymap(lambda x:'white-space:nowrap')
display(lasso_model_equation_df)
print(math.floor(x_poly_features.min()))
print(math.ceil(x_poly_features.max()))

x_test_features = []
grid = np.linspace(-4, 4)
for i in grid :
    for j in grid :
        x_test_features.append([i, j])
x_test_features = np.array(x_test_features)
poly_features_function = PolynomialFeatures(5)
x_poly_test_features  = poly_features_function.fit_transform(x_test_features)

for key in lasso_model_dictionary :
    Lasso_Model = lasso_model_dictionary[key]
    prdctns = Lasso_Model.predict(x_poly_test_features)

    figure = plot.figure(figsize=(8, 8), dpi = 80)
    plot_final = figure.add_subplot(111, projection = '3d')
    plot_final.plot_trisurf(x_test_features[:,0], x_test_features[:,1], prdctns, alpha = 0.5, color = "blue")
    plot_final.scatter(x1, x2, y, color = "green", marker = "o")
    plot_final.set_xlabel('x_1')
    plot_final.set_ylabel('x_2')
    plot_final.set_zlabel('y')
    plot_final.set_title('Data Plot for Actual VS Predicted value of y for Lasso Regression with Penalty Score: {}'.format(key))
    handles, labels = plot.gca().get_legend_handles_labels()
    prediction_legend = mpatches.Patch(color='blue', label = "Predicted Label")
    test_legend = Line2D([0], [0], color='green', linestyle="none", label = "Test Label", marker = "o")
    handles.extend([prediction_legend, test_legend])
    plot_final.legend(handles = handles, title="Legends", loc = 'lower right', bbox_to_anchor=(1.4, 0.6))
    plot_final.view_init(azim = 60)


ridge_model_dictionary = {}

ridge_model_params_df = pd.DataFrame(columns = feature_names)
ridge_model_equation_df = pd.DataFrame(columns = ['Penalty', 'Model Equation'])
for penalty in penalty_parameters:
    ridge_model = Ridge(alpha = 1 / (2 * penalty))
    ridge_model.fit(x_poly_features, y)

    model_dict = {}
    model_dict['Penalty'] = penalty
    model_dict['Intercept'] = [np.around(ridge_model.intercept_, decimals = 3)]
    for i in range(2,23) :
        model_dict[feature_names[i]] = [np.around(ridge_model.coef_[i-2], decimals = 3)]

    ridge_model_params_df = ridge_model_params_df.append(model_dict, ignore_index = True)

    model_eq_dict = {}
    model_eq_dict['Penalty'] = penalty
    equation_string = ''
    for i in range(2,23) :
        coeff = np.around(ridge_model.coef_[i-2], decimals = 3)
        if coeff != 0 :
            equation_string += '(' + str(coeff) + ')' + ' * ' + feature_names[i] + ' + '

    equation_string += '(' + str(np.around(ridge_model.intercept_, decimals = 3)) + ')'
    model_eq_dict['Model Equation'] = equation_string

    ridge_model_equation_df = ridge_model_equation_df.append(model_eq_dict, ignore_index = True)

    ridge_model_dictionary[penalty] = ridge_model

ridge_model_params_df = ridge_model_params_df.style.applymap(lambda x:'white-space:nowrap')
display(ridge_model_params_df)

ridge_model_equation_df = ridge_model_equation_df.style.set_properties(**{'text-align': 'left'})
ridge_model_equation_df = ridge_model_equation_df.applymap(lambda x:'white-space:nowrap')
display(ridge_model_equation_df)

for key in ridge_model_dictionary :
    Ridge_Model = ridge_model_dictionary[key]
    prdctns = Ridge_Model.predict(x_poly_test_features)

    figure = plot.figure(figsize=(8, 8), dpi = 80)
    plot_final = figure.add_subplot(111, projection = '3d')
    plot_final.plot_trisurf(x_test_features[:,0], x_test_features[:,1], prdctns, alpha = 0.5, color = "blue")
    plot_final.scatter(x1, x2, y, color = "green", marker = "o")
    plot_final.set_xlabel('x_1')
    plot_final.set_ylabel('x_2')
    plot_final.set_zlabel('y')
    plot_final.set_title('Data Plot for Actual VS Predicted value of y for Ridge Regression with Penalty Score: {}'.format(key))
    handles, labels = plot.gca().get_legend_handles_labels()
    prediction_legend = mpatches.Patch(color='blue', label = "Predicted Label")
    test_legend = Line2D([0], [0], color='green', linestyle="none", label = "Test Label", marker = "o")
    handles.extend([prediction_legend, test_legend])
    plot_final.legend(handles = handles, title="Legends", loc = 'lower right', bbox_to_anchor=(1.4, 0.6))
    plot_final.view_init(azim = 60)
```

Appendix: Code for Question 2 - All Parts

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plot
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from IPython.display import display
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error

df = pd.read_csv("data_week3.csv", names=["x1", "x2", "y"], header=None, comment='#')
display(df.head())
x1 = df.iloc[:,0]
x2 = df.iloc[:,1]
x = np.column_stack((x1, x2))
y = df.iloc[:,2]

figure = plot.figure(figsize=(8, 8), dpi = 80)
plot_final = figure.add_subplot(111, projection = '3d')
plot_final.scatter(x1, x2, y)
plot_final.set_xlabel("x_1")
plot_final.set_ylabel("x_2")
plot_final.set_zlabel("y")
plot_final.set_title('Data Plot for Data ID:21--42-21')
plot_final.view_init(azim = 60)

poly_features_function = PolynomialFeatures(5)
x_poly_features  = poly_features_function.fit_transform(x)


def calculate_mse_stddev_penalty_lasso(penalty_parameters) :
   k_fold_split = 5
   k_fold_split_function =  KFold(n_splits = k_fold_split)
   mean_sqaure_error_penalty = []
   standard_deviation_penalty = []

   for penalty in penalty_parameters :
      lasso_model = Lasso(alpha = 1 / (2 * penalty))
      mean_sqaure_error_fold = []
      for train_data_index, test_data_index in k_fold_split_function.split(x_poly_features):
         lasso_model.fit(x_poly_features[train_data_index], y[train_data_index])
         predictions = lasso_model.predict(x_poly_features[test_data_index])
         mean_sqaure_error_fold.append(mean_squared_error(y[test_data_index], predictions))
      mean_sqaure_error_penalty.append(np.array(mean_sqaure_error_fold).mean())
      standard_deviation_penalty.append(np.array(mean_sqaure_error_fold).std())
   return mean_sqaure_error_penalty, standard_deviation_penalty

penalty_parameters = [1, 5, 10, 100, 500, 1000]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_lasso(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Lasso Regression')
plot.show()

penalty_parameters = [1, 5, 10, 50, 75, 100]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_lasso(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Lasso Regression')
plot.show()

penalty_parameters = [1, 5, 10, 20, 30, 50]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_lasso(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Lasso Regression')
plot.show()

penalty_parameters = [1, 3, 5, 7, 9, 10, 20]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_lasso(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Lasso Regression')
plot.show()
```

```
def calculate_mse_stddev_penalty_Ridge(penalty_parameters) :
    k_fold_split = 5
    k_fold_split_function =  KFold(n_splits = k_fold_split)
    mean_sqaure_error_penalty = []
    standard_deviation_penalty = []

    for penalty in penalty_parameters :
        ridge_model = Ridge(alpha = 1 / (2 * penalty))
        mean_sqaure_error_fold = []
        for train_data_index, test_data_index in k_fold_split_function.split(x_poly_features):
            ridge_model.fit(x_poly_features[train_data_index], y[train_data_index])
            predictions = ridge_model.predict(x_poly_features[test_data_index])
            mean_sqaure_error_fold.append(mean_squared_error(y[test_data_index], predictions))
        mean_sqaure_error_penalty.append(np.array(mean_sqaure_error_fold).mean())
        standard_deviation_penalty.append(np.array(mean_sqaure_error_fold).std())
    return mean_sqaure_error_penalty, standard_deviation_penalty

penalty_parameters = [1, 5, 10, 100, 500, 1000]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_Ridge(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Ridge Regression')
plot.show()

penalty_parameters = [1, 5, 10, 50, 75, 100]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_Ridge(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Ridge Regression')
plot.show()

penalty_parameters = [1, 5, 10, 20, 30, 50]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_Ridge(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Ridge Regression')
plot.show()

penalty_parameters = [1, 3, 5, 7, 9, 10]
mean_sqaure_error, standard_deviation = calculate_mse_stddev_penalty_Ridge(penalty_parameters)
plot.figure()
plot.errorbar(penalty_parameters, mean_sqaure_error, yerr = standard_deviation, color = 'blue')
plot.xlabel('Penalty')
plot.ylabel('Mean square error')
plot.title('Plot of MSE VS Penalty for 5-fold Cross Validation Ridge Regression')
plot.show()
```