Machine Learning Diagnostics:

In this report, three Machine Learning Diagnostics were used to try to develop a better fitting Linear Regression model for a house pricing dataset. Four different hypothesizes were tested on each algorithm to check which one gave the best result for the model.

Feature Selection:

Before applying Machine Learning algorithms, filtering of the features to get the most important ones was applied in order to lower the complexity of the models. Feature Selection was divided into two parts: Feature elimination, and choosing features that could benefit from applying polynomial degrees on them.

Feature Elimination:

The .corr() function was applied on the dataset to get the correlation between the features and then eliminate a feature if a correlation of 0.5 or higher between it and another feature was found, this was done to eliminate redundant features, both to reduce the complexity of the model and to avoid overfitting the model.

Feature selection for Polynomial application:

After the removal of redundant features, five features were selected to be used for polynomial hypothesizes. This was done by first; choosing feature columns with non-binary values, as adding a polynomial to these features will not be of any benefit, and second; plotting the rest of the features against the target (i.e the price) to select features that could help the model if a polynomial of it was added to the dataset.

After selecting the features suitable for polynomial regression, the function MakePoly(X,degree) was built so that upon calling, it returns a new X with polynomial features of the chosen degree, an example execution is shown Below.

Input	Output
MakePoly(X,2) $X = x_1 + x_2$	$X = x_1 + x_2 + x_1^2 + x_2^2$

Diagnostics:

Linear Regression using K-fold Sampling:

K-fold sampling splits the data into n subsets and then proceeds to train on n-1 subsets and tests on one, in a cyclic manner until all of the data is used both for training and for testing. This was done by using the built-in function KFold(n_splits = n). then, to build the model, three functions were developed to first, normalize the data, then compute the cost $J(\theta)$ according to eq(1) below, and then apply gradient descent eq(2) to optimize the function. After applying the original dataset on the model, second, third, and fourth degree polynomial hypothesizes were also applied to see which one gave the least root mean squared error between the predicted data and the actual data. Upon noticing, the degree 2 polynomial hypothesizes gave the best result (defined by the least root mean squared error).

$$J(\theta_0, \theta_1) = 0.5 * m * \sum_{i=1}^{m} (h\theta(x^i) - y^i)^2 \; eq(1)$$

$$\theta_j = \theta_j - \frac{\alpha}{m} * \sum_{i=1}^{m} (h\theta(x^i) - y^i)^2 \; . x_j^i (2)$$

Linear Regression using StratifiedK-fold Sampling:

Stratified sampling ensures the data selected has the same ratio of classes to be a better representative of the data. The same steps for K-fold sampling were followed, however the function StratifiedKFold(n_splits = n) was used to ensure a fair representation of the data classes was selected. After testing different alphas on each of the four hypothesizes, the degree polynomial 2 with alpha 0.03 yielded the best result, based on the least root mean squared error metric. However, K-fold sampling had an overall better result on the model.

Linear Regression using Regularization:

Lastly, regularization was applied to the model to find the lowest complexity that would give an optimal fit for the data. Regularization works by adding a penalty term to the cost function of the model that increases by increasing the complexity. For the testing process, twelve different lambdas were used to check which one would work best with the data, along with four different hypothesizes (degree 1, degree 2, degree 3 and degree 4).

To develop the regularization model, two extra functions were built:

1. RegcomputeCostMulti(X, y, theta,lambd)
2. RegradientDescentMulti(X, y, theta, alpha, num_iters,lambd)

These two functions apply the same equations for the normal linear regression model. However, they add a regularization term to it to compute the penalty in each iteration, the equations are shown in eq(2) and eq(3) below. The data was split in a 60:20:20 ratio where 60% were assigned to the training data, 20% for the cross-validation data and 20% for the test data.

The RegradientDescentMulti(X, y, theta, alpha, num_iters,lambd) function was used to compute the thetas derived using each of the twelve lambdas (except for θ_0, $which\ was\ computed\ without\ adding\ the\ penalty\ term$), then the resulted set of thetas were used to calculate the cost on the cross-validation dataset, without adding the penalty term. The cost of using each of the theta-lambda combination was saved in an array, where the index of each of them represented the lambda used.

The minimum cost was then calculated and the theta-lambda combination that resulted in it was applied to the test data to see if it was a good fit. However, that combination did not result in a small margin of error for the test data, suspecting that the model was under fitted on the validation set, as the lambda chosen was considerably large in size. The process was repeated for each of the hypothesizes, leading to the same result, leading that overall, K-fold sampling has led to the best fitting of the data.

$$J(\theta_0, \theta_1) = 0.5 * m * \sum_{i=1}^{m} (h\theta(x^i) - y^i)^2 \ + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta^2 \ eq(2)$$

$$\theta_0 = \theta_0 - \frac{\alpha}{m} * \sum_{i=1}^{m} (h\theta(x^i) - y^i)^2$$

$$\theta_j = \theta_j \left(1 - \frac{\alpha\lambda}{m}\right) - \frac{\alpha}{m} * \sum_{i=1}^{m} (h\theta(x^i) - y^i)^2 \, . x_j^i (3)$$