Omar Emam 40-8001

**ML Assignment 1**

**Objective:**

Given a set of data in a csv file describing real estate properties features we should implement an algorithm to predict the price of a real estate after training the model upon the already given features.

**Preparing the data:**

The data is composed of a target variable which is the price “label” and all the other columns are considered as features, the objective is to predict the label by applying a non changing model, the data should be prepared in a way to facilitate a more precise prediction by implementing a set of operations.

**Dropping uncorrelated data with label:** Applying .corr() function produces 2D array of numbers showing how dependent is each field is on every other field so the point of focus will be the label, after determining a certain threshold of correlation, every feature below the threshold determined should be dropped to allow a more accurate prediction of the label so that the uncorrelated features do not interfere with the calculations.

**High degrees of very correlated features:** After checking the correlated values picking the higher ones to up the degree of them by 2 or 3 or more can make the prediction accuracy better due to the fact that the more dependent values are now more effective function-wise

**Dividing the data into sets:**

Data is divided into 3 sets normally with 60% of the data corresponding to the data in which the model is trained then 20% of data is used for cross validation and 20% of the data is used for test data.

**K-Fold sampling:** Data is divided into K sub-samples where one sample of them is taken to be the test sample and the other K-1 data is taken to be the train sample then an iteration occurs where the test sample is switched with each one of the other K-1 data and new test sample is introduced and new training sample as well on each iteration.

**Regularization:**

Regularization is used on data to reduce parameters on the features to avoid overfitting, overfitting of data is not healthy for the prediction most of the times on most of the data samples especially the more varying complex ones so regularization can be used as means of improving accuracy on such data sets.

**Assignment 1:**

**Preamble:** First the csv file of real estate properties is being read and placed into a variable called “data” with all values of nan being dropped then a “label” variable is initialized and it takes the entire column of prices in the csv and another variable called “features” takes all the other columns

**Date Manipulation:** Two cells of data manipulation are introduced one in which drops certain columns in the data and the other provides certain variables with a higher degree. **How to decide?** .corr() function is invoked on the data producing correlations of columns with each other, the point of focus here should be the correlation of the price column with all the other columns, all the values below a correlation of 0.2 are dropped in the cell responsible and by trial and error the values in which I invoked a higher degree upon them are initially determined by high correlation and then by trying to swap them. This is the final product:

features['bedrooms2'] = features['bedrooms']\*\*2

features['sqft\_basement2'] = features['sqft\_basement']\*\*2

features['lat2'] = features['lat']\*\*2

features['view3'] = features['view']\*\*3

features['bathrooms4'] = features['bathrooms']\*\*4

features['sqft\_living154'] = features['sqft\_living15']\*\*4

features['grade5'] = features['grade']\*\*5

features['sqft\_above5'] = features['sqft\_above']\*\*5

features['sqft\_living6'] = features['sqft\_living']\*\*6

Then all the lists are translated into numpy lists to facilitate access of data. However, one problem might present itself is the existence of very large numbers so it is avoided by invoking a normalization of data cell

**Data splitting:** Firstly, data is splitted into 60% training set, 20% cross validation and 20% test then these sets of data are invoked on the function gradientDescentMulti the function takes the features, the label and a theta of the same length as the features and the label and a tuning alpha and num\_iter to produce the output thetas in which the cost will be computed upon.

**Test prediction results:** The function computeCostMulti computes the costs of prediction the costs are as following:

* A trend of lower cost if non correlated data is dropped and a trend of lower cost if polynomials are applied.
* Highest cost is when data is dropped and no polynomials are applied
* Applying k-fold with a single iteration produces better test results than pure non drop non poly but worse than drop and poly combined
* If lambdas regularization method is applied the best iteration result produces a similar result to dropping and poly or slightly better which means no overfitting is existent in the case.

**Assignment 2:**

The data consists of 3 fields in which the first two is used to predict the 3rd one .

**Preamble:** features is considered to be the first two columns and the label is the 3rd one, the nature of the label is binary meaning that there can only be two possible outputs 1 or 0.

**Data Manipulation:** Both of the features correlate with the output so it makes no sense to drop any of them but, the first column correlates more than the second one so a higher degree is introduced to the first column.

**Data splitting:** Data is split into 60% train 20% cross validation and 20% test.

**Prediction:** The given cost function produces a logistic regression based output in which is used with scipy library which is a library for optimization to compute the minimal cost of the set of predictions then the minimal cost function is given to the predict functions in order to carry out the prediction, regularization is found to be inefficient.

**Test prediction results:** The highest results are found when no data manipulation is done and the functions are left to predict on the raw data, producing prediction results ranging from 85% to 100%.