CS6510: Applied Machine Learning

Assignment 3

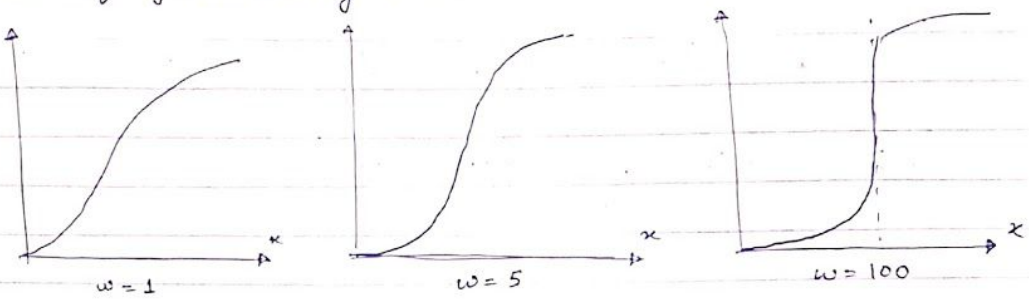
Saksham Mittal

18.04.2019

CS16BTECH11032

Theory Questions

(1) (a) Plots of sigmoid change with $w \in \{1, 5, 100\}$



$w = 1$ $\quad$ $w = 5$ $\quad$ $w = 100$

On increasing w, the curve gets steeper. Steeper curve means that model predicts with almost 0 or almost 1 probability. When the weights are large, even small change in input can change the probability of class which can change the prediction.

(b) MAP estimation is given by:

$$\max_{w_0, \ldots w_d} \prod_{i=1}^{n} P(Y_i \mid X_1, w_0, \ldots, w_d)\, P(w_0, w_1, \ldots w_d)$$

We first take the log of conditional posterior

$$w = [w_0, w_1, \ldots w_d]^T$$

$$L(w) = \log \left(p(w) \prod_{j=1}^{n} P(y_j \mid x_j, w) \right)$$

Since the prior is a standard Gaussian $N(0,1)$

$$p(w) = \prod_{i=0}^{d} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{w_i^2}{2} \right)$$

Now, the maximum conditioned a posterior estimate

$$\underset{w}{\operatorname{argmax}} \left[\sum_{j=1}^{n} \log\left(P(y^i \mid x^i, w) \right) - \sum_{i} \frac{w_i^2}{2} \right]$$

To find the gradient ascent update rule, we need to find

$$\frac{\partial L(w)}{\partial w_i} = \frac{\partial}{\partial w_i} \log p(w) + \frac{\partial}{\partial w_i} \log \left(\pi P(y^i \mid x_i, w) \right)$$

So, the final update rule is:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left(-w_i^{(t)} + \sum_j x_i^j \left(y^j - P(Y=1 \mid x^j, w^{(t)}) \right) \right)$$

{ using the result: $\frac{\partial}{\partial w_i} \log(p(w)) = -w_i$ }

(c) Since, sum of probabilities of all classes = 1.

$$P(Y=y_K \mid X) = 1 - \sum_{i=1}^{1} P(Y=y_i \mid X)$$

We can define, $P(Y=y_k \mid X) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp(w_{k_0} + \sum_{j=1}^{d} w_{ij} X_j)}$

And for $k = 1, 2, 3 \ldots K-1$

$$P(Y=y_k \mid X) = \frac{\exp(w_{k_0} + \sum_{j=1}^{d} w_{kj} X_j)}{1 + \sum_{i=1}^{K-1} \exp(w_{k_0} + \sum_{j=1}^{d} w_{ij} X_j)}$$

we can verify that $\sum_{k=1}^{K} P(Y=y_k \mid X) = 1$

↘ Hence, the definition is correct

The classification rule is picking the label with highest probability:

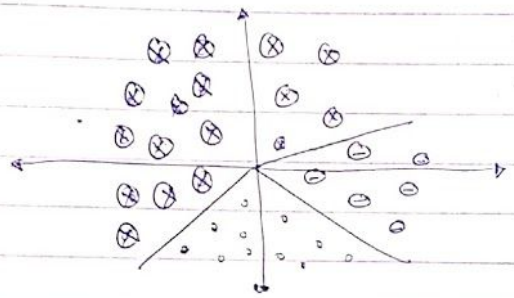
$$y = \underset{k \in \{1,2..K\}}{\operatorname{argmax}} \; P(Y=y_k \mid X)$$

(d) For classification, we need $\underset{k}{\operatorname{argmax}} \; P(Y=y_k \mid X)$

$\Rightarrow \underset{k}{\operatorname{argmax}} \left(\exp\left(w_{k_0} + \sum_{i=1}^{d} w_{ki} X_i\right) \right)$ [from part (c)]

$\Rightarrow \underset{k}{\operatorname{argmax}} \left(w_{k_0} + \sum_{i=1}^{d} w_{ki} X_i \right)$ { $\because$ exp() is an increasing function }

since max of linear functions is peice wise linear, the overall decision boundary is peice wise linear.



(2) (a) In kernel regression, the output is given as:

$$y^T (G + \lambda I)^{-1} \bar{z}$$

where G is the Gram matrix

and $\bar{z} = \langle \phi(x_i), \phi(x) \rangle$

$$\Rightarrow \bar{z} = K(x_i, x) = \exp\left(\frac{||x_i - x||_2^2}{\sigma^2}\right)$$

$$\therefore \hat{y} = y^T (G + \lambda I)^{-1} \exp\left(\frac{||x_i - x||_2^2}{\sigma^2}\right)$$

{ on substituting }

$$\Rightarrow \hat{y} = y^T l(\bar{x}) \quad \text{where } l(\bar{x}) = (G + \lambda I)^{-1} \exp\left(\frac{||x_i - x||_2^2}{\sigma^2}\right)$$

$$\therefore \boxed{\hat{y} = l(\bar{x})^T \cdot y^T}$$

So, Kernel regression is a linear smoother.

(b) It will not be a linear smoother anymore. Consider the example: Each of the training input instance has $x_i = 1$ for different y values. Then w will be the median of the y's. We can see that $\bar{w}$ is not linear in any of the y's, as the median value changes as rank of y's does.

(c) Yes the estimate will be a linear smoother, the vector $l(x)$ can be given as:

$x \in B_i$, $\quad l_j(x) = \dfrac{I(x_j \in B_i)}{|B_i|}$

$x \notin B_i$ $\quad l_j(x) = 0$

Programming Questions

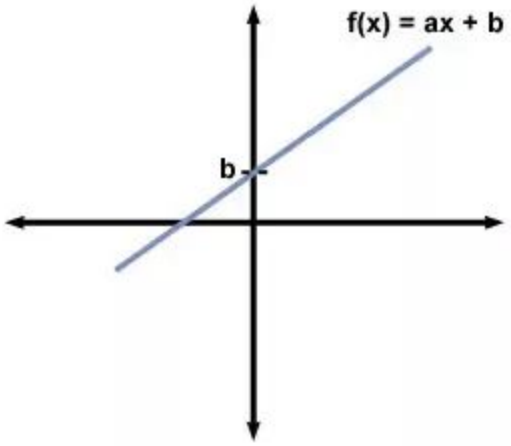
[Python version used in assignment = 2.7.15]

3. Linear Regression:

I have used Pandas and Numpy for data preprocessing. The csv is read using Pandas otherwise, the string constants were showing nan.

For one hot encoding, I have used the Pandas library's get_dummies() method, and concatenates the result to the original data. Then we normalize the data also, by subtracting it with its mean and dividing it with its standard deviation.

An extra column of ones is also appended to the data, which is the bias of the linear regression model. (As shown in the figure)



The functions(mylinridgereg(X, Y, λ), mylinridgeregeval(X, weights), meansquarederr(T, Tdash)) involves matrix multiplication and inverses, so I used numpy for these calculations.

For splitting the data in training and test set, I used sample() method from the Pandas library.

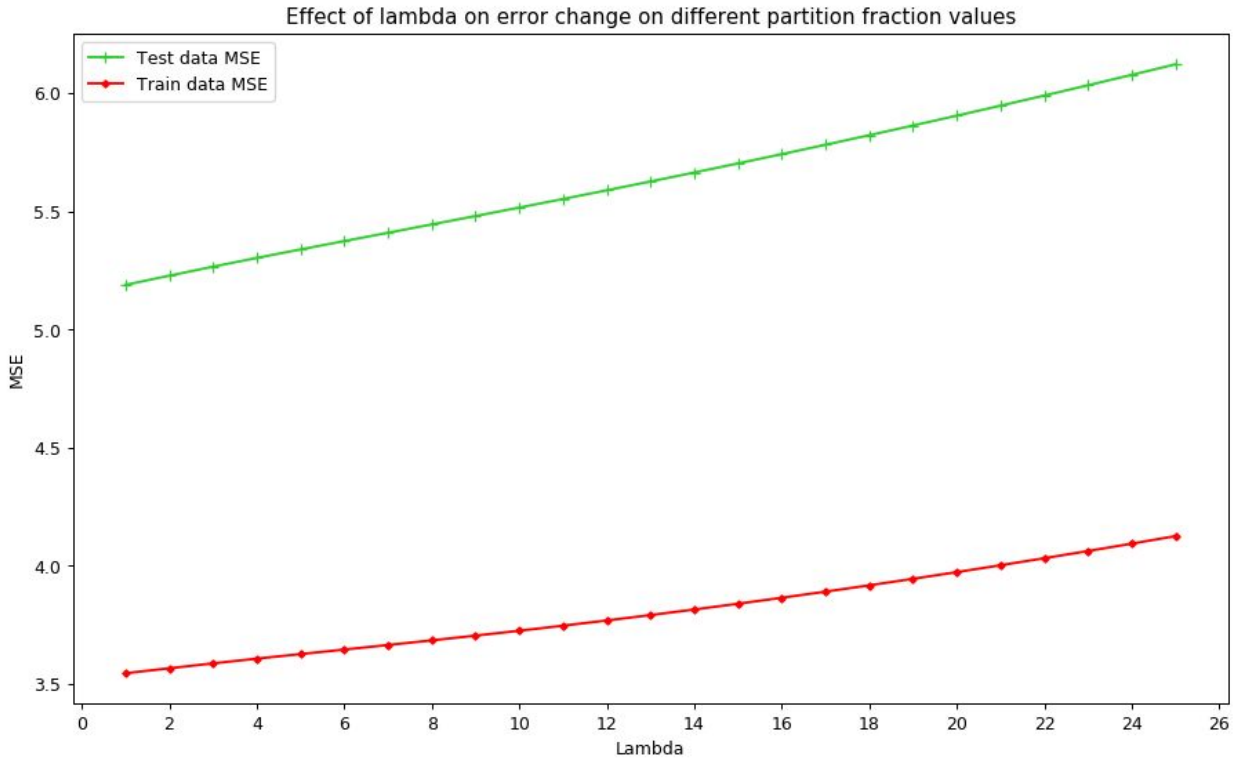
I used the following lambda values: 0.01, 0.1, 1, 10, 100 for finding various test and train target variable.

The lambda with the best performance comes out to be 10 on an average of 5 tries.

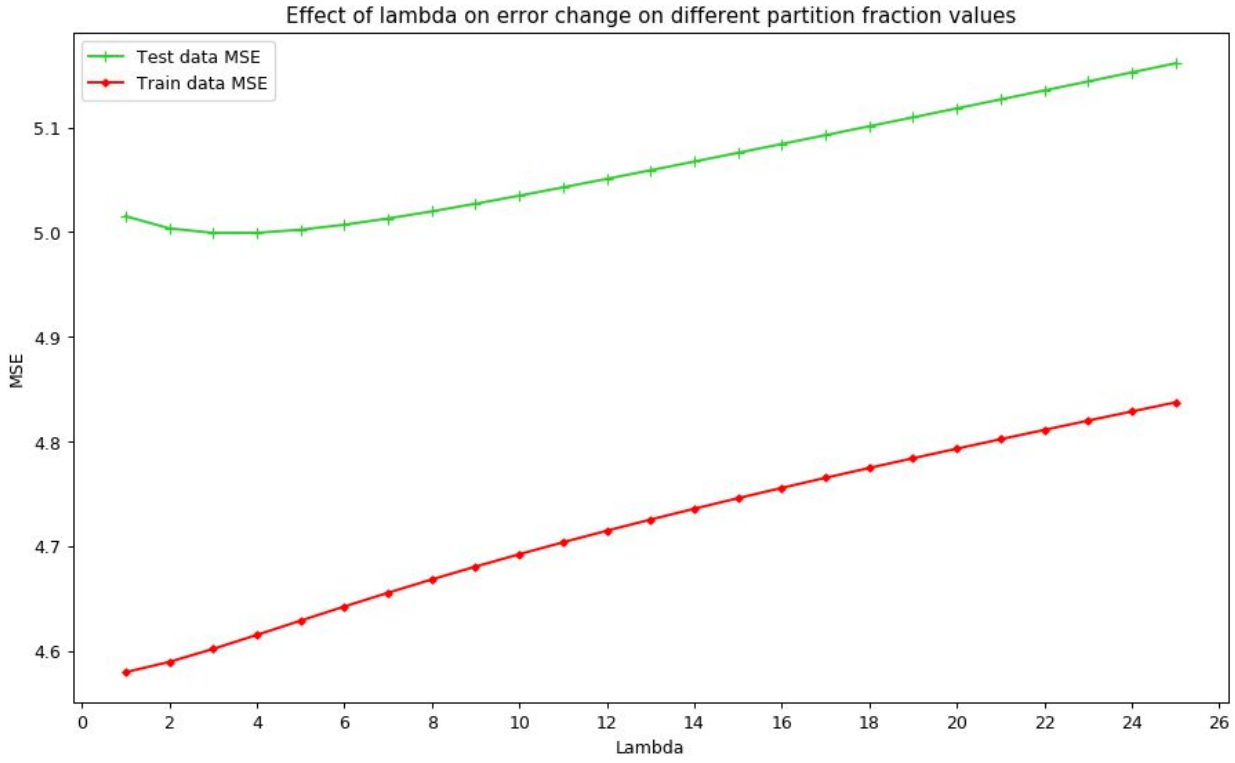
For removing the least significant attributes, I used the absolute values of the predicted

weights, and removed the least 2 from them. The Average MSE increases after removing the least 2 significant attributes.

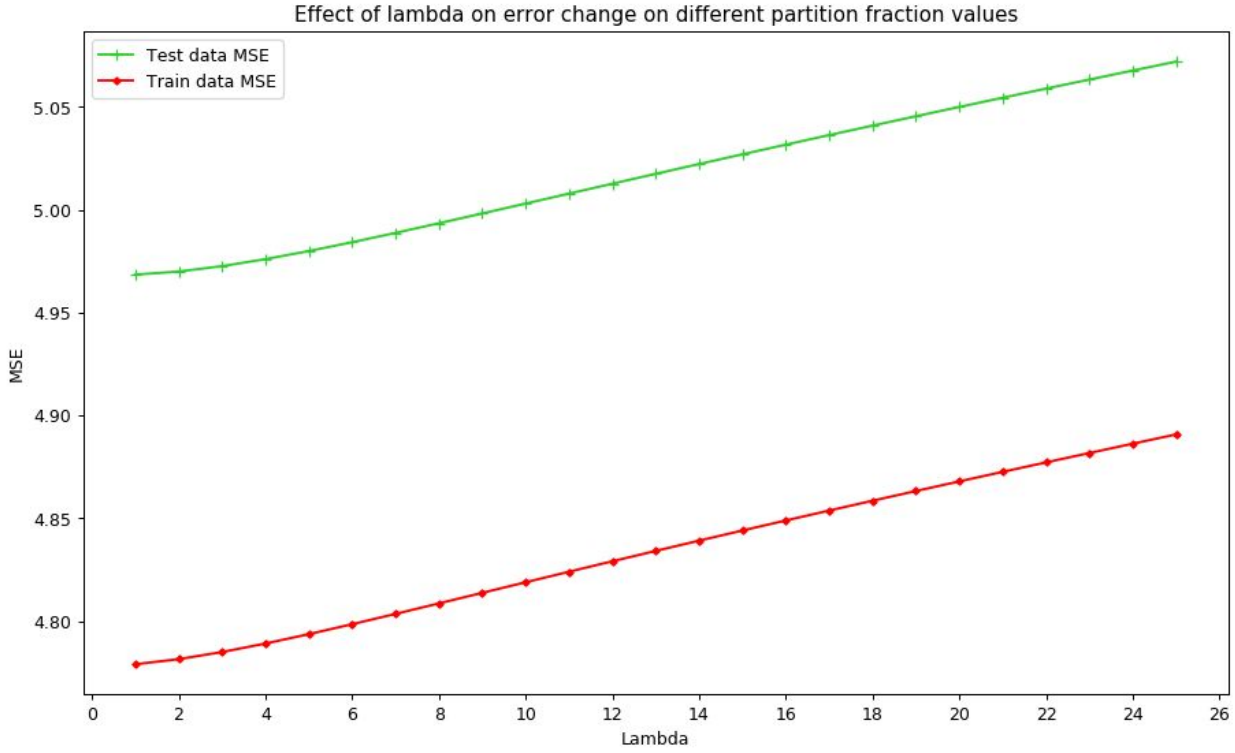
Plots for part (f)



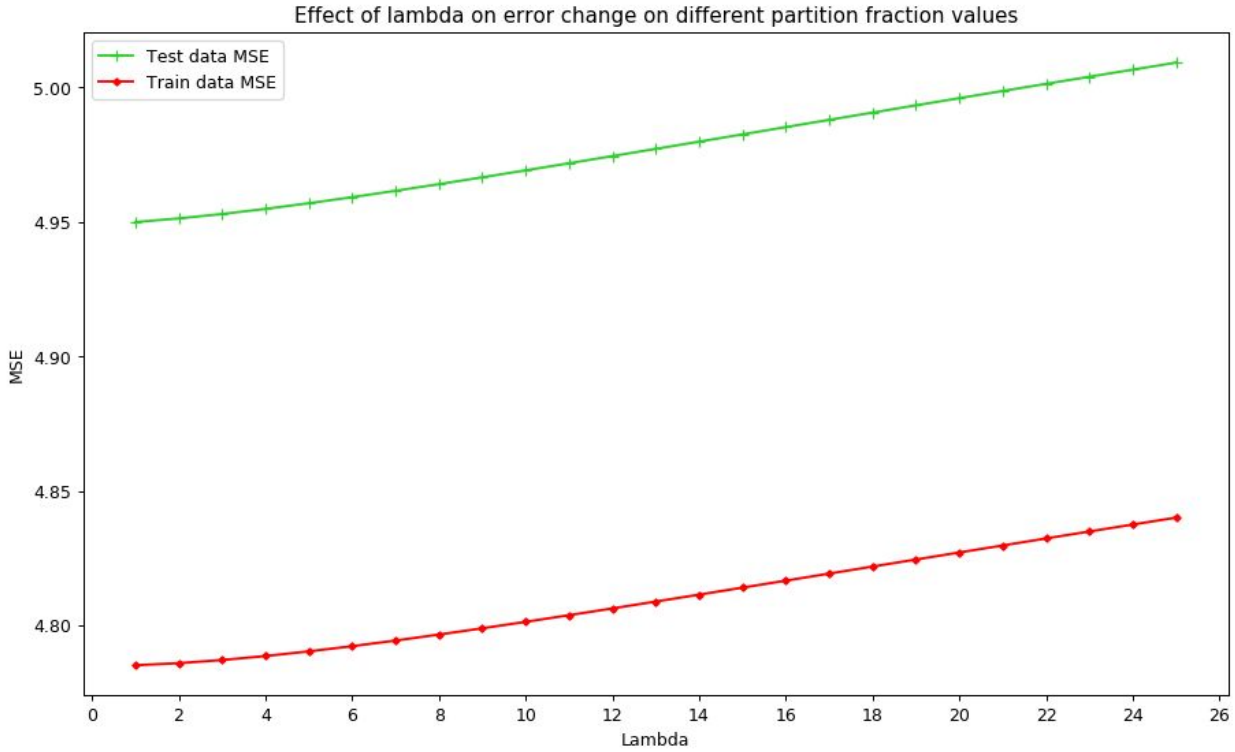
F.1 *For the fraction 0.1*



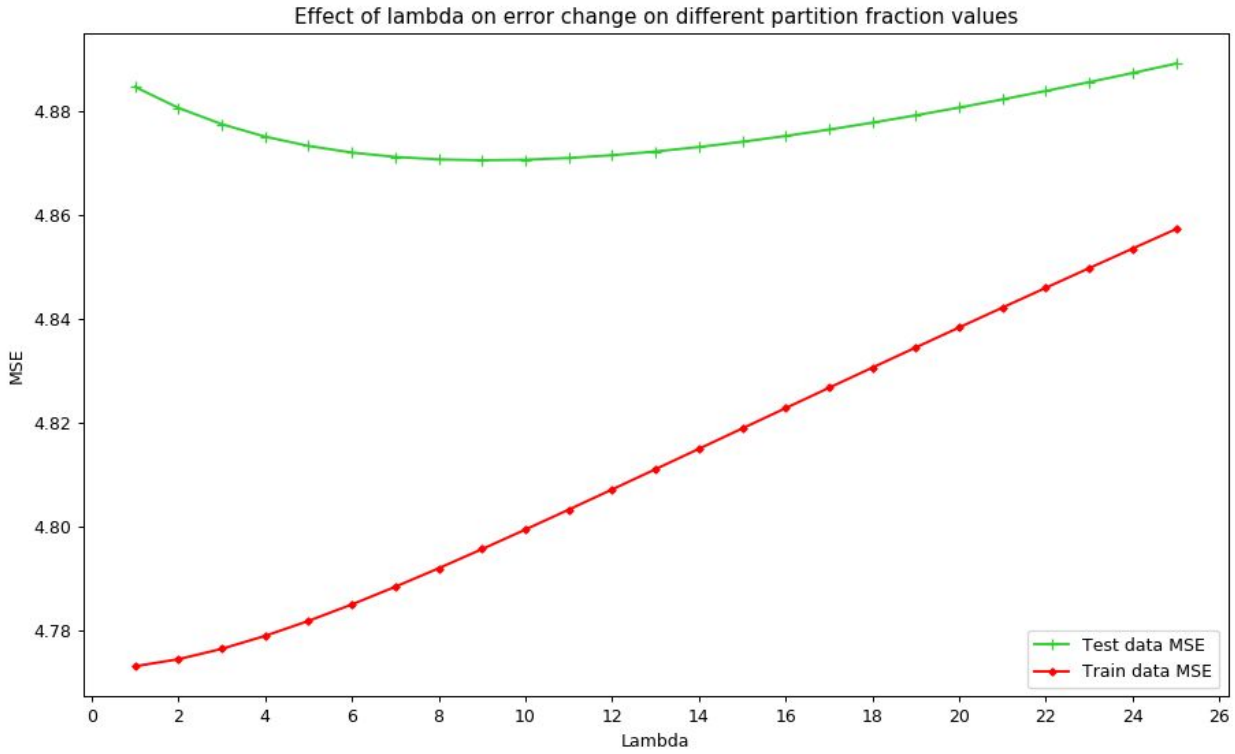
F.2 For the fraction 0.3



F.3 For the fraction 0.5



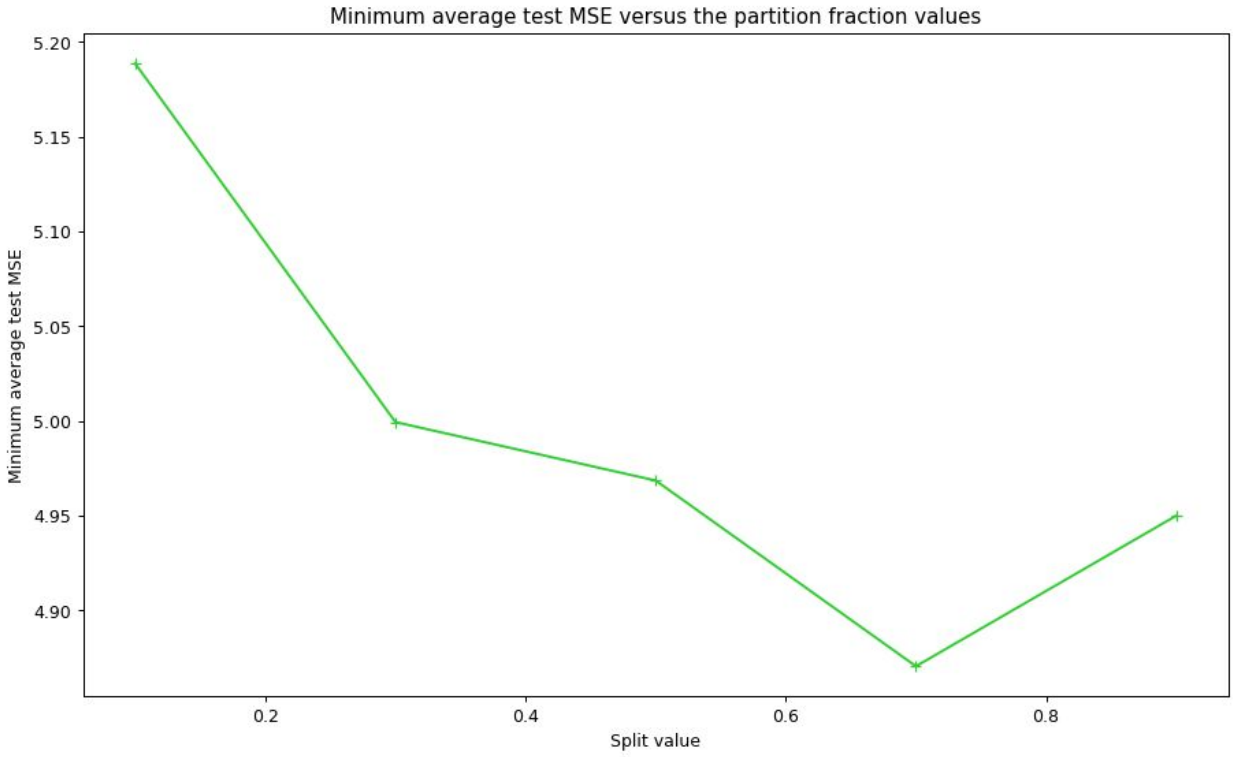
F.4 For the fraction 0.7

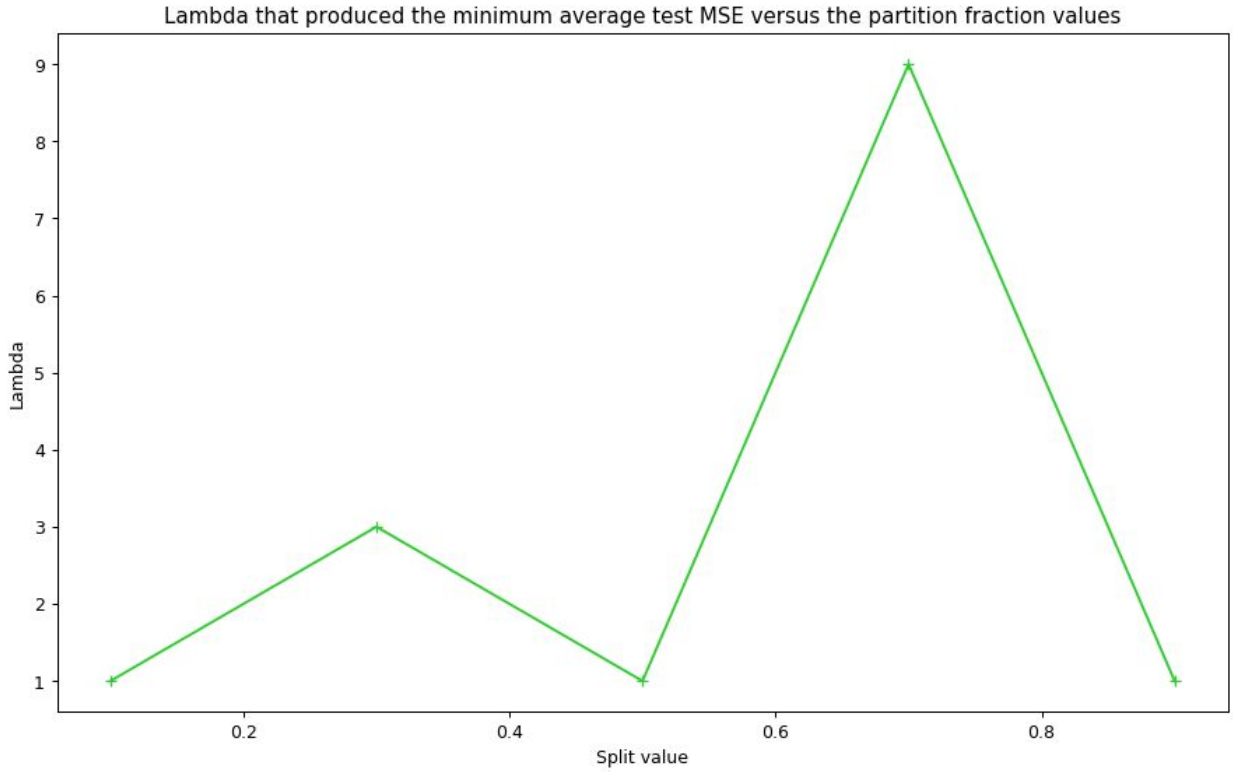


F.5 For the fraction 0.9

The test error decreases on increasing the fraction because the model is trained on more samples now. Also, error(both training and test) increases on increasing lambda.

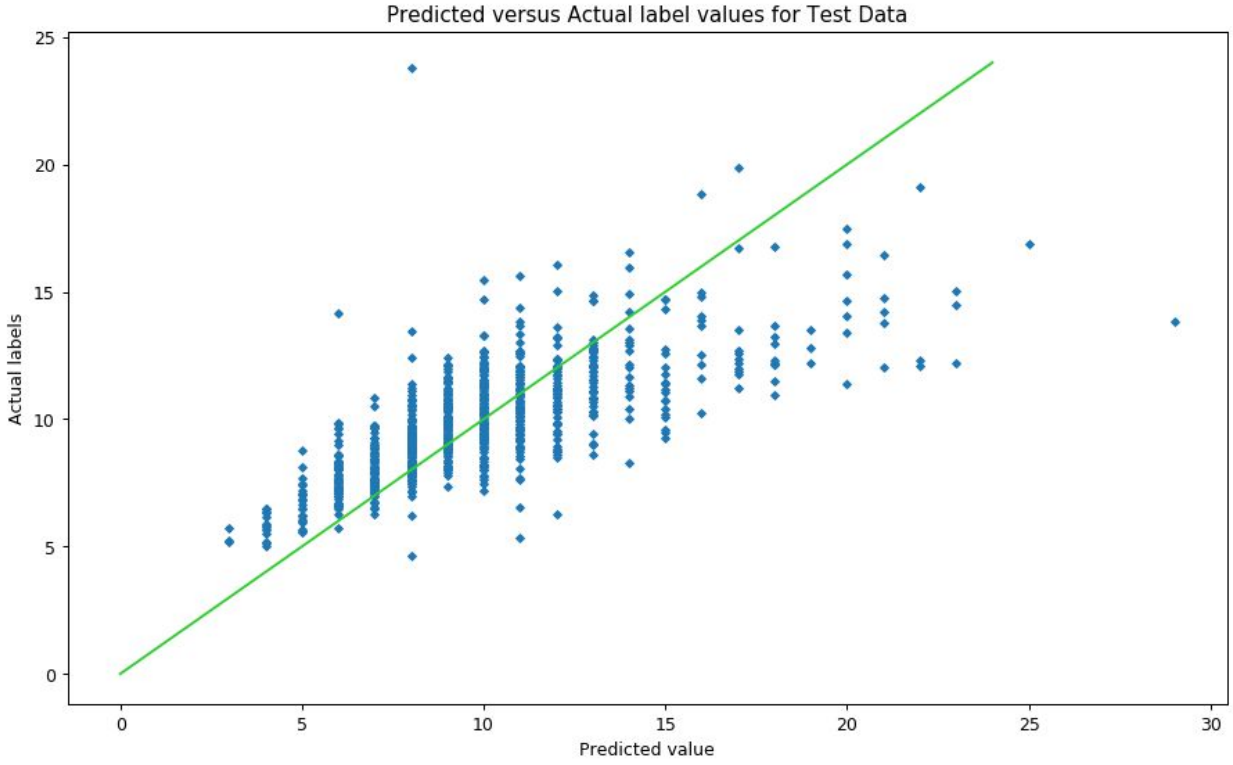
Plot for part (g)

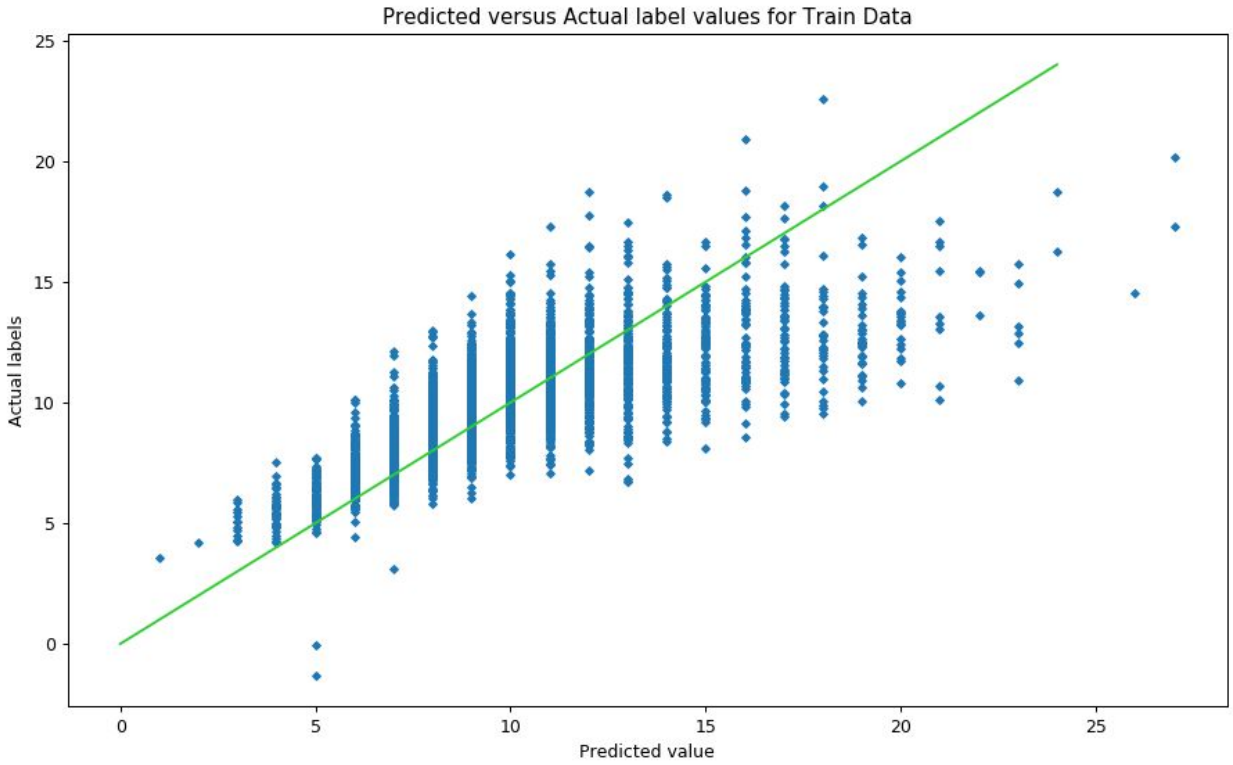




As we can see the split value for the minimum average Test MSE is obtained at ~ 0.7 for the lambda ~ 9.

Plots for part (h)





We can see that the values lie close to the green 45-degree line.

The code for parts a), b), and c) are provided in 3-linear-regression.py

The code for parts d), e) is provided in 3d-linear-regression.py

Rest are provided in 3f-linear-regression.py, 3g-linear-regression.py and 3h-linear-regression.py

4. Kaggle - Taxi Fare Prediction:

Data preprocessing:

I have used pandas for handling CSV file I/O and numpy for mathematical linear algebra.

Because of limited memory on my local machine, I used only first 1000000 rows of train data for predicting the test data. For removing outliers, I removed rows which had any **nan** values, which were a small amount.

Then, I added a new feature '**distance'** which was calculated from pickup_latitude, pickup_longitude, dropoff_latitude, and dropoff_longitude. I used the Haversine formula for this calculation. Then I dropped those 4 features. This helped to compress information in just a single feature.

Then, I parsed the pickup_datetime feature in year, month, day, hour, and minute features. I ignored the second feature because it doesn't affect the fare amount much.

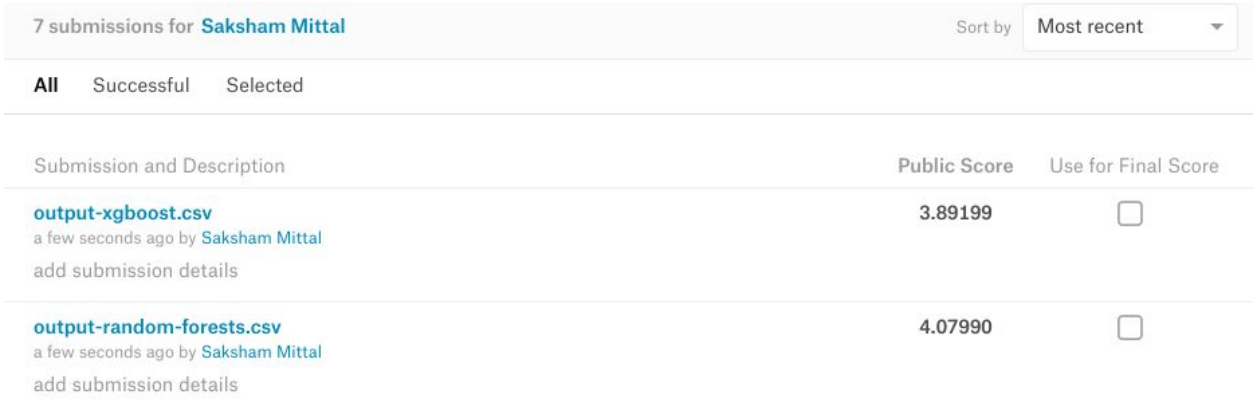
I also removed fare_amount from the training data and kept in a separate training_labels numpy array.

The same preprocessing was done to test data as well.

Results:

My top-2 scores came from Random Forest regressor and XgBoost regressor.

The scores were:



For random forest regressor, I used 50 estimators for the training. The objective method for the xgboost regressor was default.

The code is provided in 4-kaggle.py file.

The reason for using xgboost and Random Forest regressor was that the data included both categorical and continuous features, which is handled well by the tree based regressors. Using gradient boosting helped to decrease MSE as we can see that MSE for xgboost is less than Random forests.