Machine, Data and Learning
Assignment 2

Name: Abhinav Reddy Boddu
Roll No: 2021101034

1 Task 1: Linear Regression

1. The method $LinearRegression().fit()$ is a function used to train a linear regression model on a given dataset.
2. It uses the ordinary least squares (OLS) method to estimate the coefficients of the linear regression model
3. The method tries to estimate the coefficients of linear model which minimize the sum of the squared residuals between the predicted values and the actual target values.

$$cost = \text{sum of squared residuals} = \sum_x (\hat{f}(x) - f(x))^2$$

Where x is the input values of data points

$\hat{f}(x)$ is the output predicted by our model for given input x

$f(x)$ is the target values for given input x

4. It is a supervised learning algorithm used to predict a continuous output variable based on one or more input variables.
5. The $fit()$ method can handle multiple input variables and can also be extended to include polynomial and interaction terms to model more complex relationships between the input and output variables.
6. $args$:
   (a) X: input or feature matrix.It is a 2-dimensional array where each row corresponds to a sample or observation, and each column corresponds to a feature or input variable. (independent Variables)
   (b) Y: This is the other mandatory argument and represents the target or output variable (Dependent Variable)
   (c) sample_weight: The sample weights are used to give more importance to certain observations in the training dataset than others. If not specified, all observations are given equal weight

2 Task 2: Gradient Descent

Gradient descent is an iterative optimization algorithm that can be used to find the coefficients or weights of a machine learning model that minimize a cost function. Where The cost function is a measure of how well the model fits the data and can be defined as the mean squared error between the predicted values and the actual values of the dependent variable.

In the case of a linear regression model with one independent variable and one dependent variable, the goal is to find the slope and intercept of the line that best fits the data.

The basic idea of gradient descent is to update the coefficients iteratively by moving them in the direction of steepest descent of the cost function.

working of gradiant descent method:

1. **Initialize the coefficients**: Start with some initial values for the slope (m) and intercept (b) of the linear regression line generally taken as m=0 and b=0.

2. **Calculate the cost function**: Calculate the mean squared error between the predicted values of the dependent variable (y_{pred}) and the actual values of the dependent variable (y_{true}) using the current values of the coefficients.

$$cost = MSE = \frac{1}{n}\sum(y_{pred} - y_{true})^2$$

3. **Calculate the gradient**: Calculate the partial derivatives of the cost function with respect to the slope and intercept.

$$\frac{d(cost)}{dm} = \frac{d(cost)}{d(y_{pred})}\frac{d(y_{pred})}{dm}$$

$$\text{Since } y_{pred} = mx + b, \frac{d(y_{pred})}{dm} = x$$

$$\text{also } \frac{d(cost)}{d(y_{pred})} = \frac{1}{n}\sum(2(y_{pred} - y_{true}))$$

$$\Rightarrow \frac{d(cost)}{d(y_{pred})} = \frac{2}{n}\sum((y_{pred} - y_{true}))$$

$$\Rightarrow gradient_m = \frac{d(cost)}{dm} = \frac{2}{n}\sum(y_{pred} - y_{true})x$$

Similarly for b,

$$\frac{d(cost)}{db} = \frac{d(cost)}{d(y_{pred})}\frac{d(y_{pred})}{db}$$

$$\text{Since } y_{pred} = mx + b, \frac{d(y_{pred})}{db} = 1$$

$$\text{also } \frac{d(cost)}{d(y_{pred})} = \frac{1}{n}\sum(2(y_{pred} - y_{true}))$$
$$\Rightarrow \frac{d(cost)}{d(y_{pred})} = \frac{2}{n}\sum((y_{pred} - y_{true}))$$
$$gradient_b = \frac{2}{n}\sum(y_{pred} - y_{true})$$

where x is the independent variable.

4. **Update the coefficients**: Update the values of the slope and intercept by moving them in the direction of steepest descent of the cost function. The size of the update is determined by the learning rate α, which is a hyperparameter that controls the step size.

$$m = m - \alpha * gradient_m$$
$$b = b - \alpha * gradient_b$$

5. **Repeat steps 2-4 until convergence**: Repeat steps 2-4 until the change in the cost function is below a certain threshold or a maximum number of iterations is reached. Convergence is reached when the coefficients no longer change significantly.

6. **Use the final coefficients**: Once the algorithm has converged, the final values of the slope and intercept can be used to make predictions on new data.

3 Task 3.2: Bias & Variance:

Degree of polynomial	Bias	Variance
1	0.269416	0.00682374
2	0.0861101	0.00130659
3	0.0335251	0.000546639
4	0.0261134	0.000969678
5	0.0261704	0.00148358
6	0.0269571	0.00256623
7	0.029167	0.00645294
8	0.0338253	0.0157929
9	0.035058	0.0240377
10	0.0296074	0.0262106
11	0.0317448	0.0574965
12	0.0478254	0.127828
13	0.249979	4.9005
14	0.611652	6.663
15	0.845048	4.103

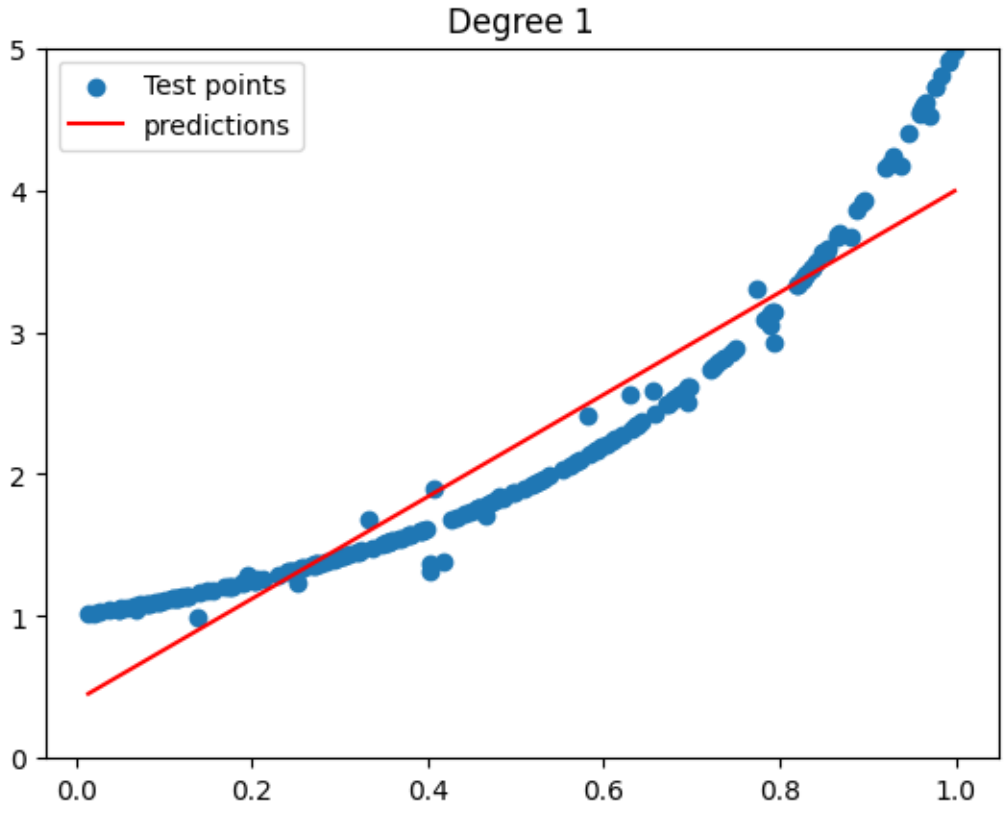
As the degree of polynomial function increases, the complexity of the model also increases. As a result, both the bias and variance of the model may change.

3.1 Simpler model(Polynomials of lower degree)

A less complex model generally does not recognize the pattern in the data i.e it underfits the data, therby predicted output of the model and the actual output (i.e Bias) differ greately.

But then recognizing less pattern means the model can fit to any general data, Thereby very low variance

Thereby a less complex model causes more bias and less Variance compared to relatively more complex model

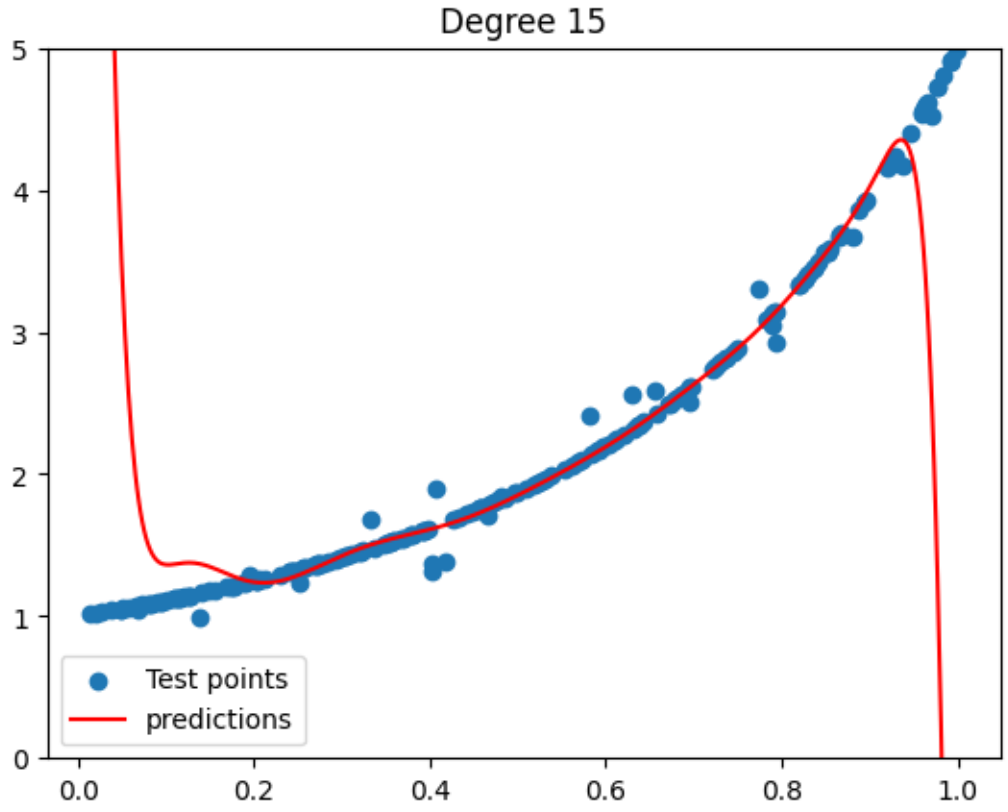


3.2 Complex models(Polynomials of higher degree)

A complex model generally does recognizes more pattern in the data than required i.e it overfits the data, therby predicted output of the model and the actual output (i.e Bias) differ by very small values.

But then recognizing more pattern means the model can't fit to any general data, Thereby higher variance from test case to test case

Thereby a more complex model causes less bias and high Variance compared to relatively simple models



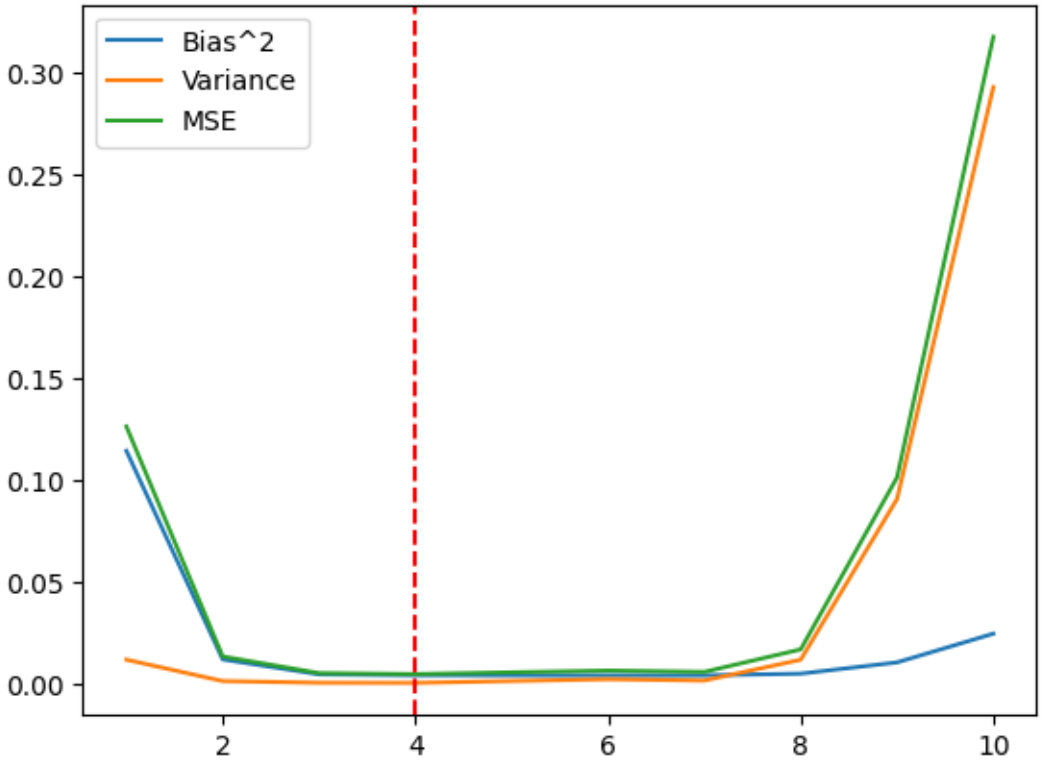
4 Task 4: Irreducible error

Degree of polynomial	Irreducible error
1	-5.19116×10^{-18}
2	-1.16498×10^{-18}
3	-1.95536×10^{-18}
4	-4.57127×10^{-19}
5	-1.36975×10^{-18}
6	2.46548×10^{-18}
7	-3.32931×10^{-18}
8	1.65883×10^{-19}
9	5.00305×10^{-18}
10	2.91713×10^{-17}
11	-1.11784×10^{-17}
12	8.90867×10^{-17}
13	6.04636×10^{-16}
14	-5.87047×10^{-15}
15	-1.39834×10^{-14}

The irreducible error is the error that cannot be reduced by any model, regardless of its complexity. It is a result of the noise inherent in the data or any other unpredictable factors that affect the dependent variable. As such, the value of the irreducible error should not change as we vary our class function, as it is not dependent on the model itself.

It is called irreducible because it cannot be eliminated by any improvements in the model

5 Task 5: Bias^2 , Variance tradeoff



Key observations of Bias^2-Variance Graphs:

1. **Underfitting**: We can observe that the polynomial models with degrees 1-2 have high bias ($underfitting$) because they are too simple to capture the complexity of the data. These models are not able to fit the training data well and have high MSE (total error). Therefore, they are not suitable for the given data.

2. **Overfitting**: We can observe that the polynomial models with degrees > 8 have high variance ($overfitting$) because they are too complex and capture the noise in the training data. These models have low bias but high MSE (total error) because they fit the training data too well and do not generalize well to new, unseen data. Therefore, they are not suitable for the given data.

3. **Good fit**: We can observe that the polynomial models with degrees 3-7 have the best balance between bias and variance and achieve the lowest MSE (total error). These models are able to capture the complexity of

the data without overfitting, and they generalize well to new, unseen data. Therefore, they are good fit models for the given data.

4. **Type of data**: The Bias^2-Variance plot suggests that the data has a non-linear relationship between the input features and the target variable, as indicated by the U-shaped curve. It also suggests that there may be some noise in the data, as indicated by the high MSE (total error) across all polynomial degrees. Therefore, it may be necessary to preprocess the data to reduce noise or transform it to a more appropriate format before fitting a polynomial regression model.

6 Bonus:

Given:

$$Q = CV_0 e^{-\frac{t}{RC}}$$

We can clearly see it is not linear.
On applying ln()

$$\ln Q = \ln(CV_0 e^{-\frac{t}{RC}}) = \ln(CV_0) - \frac{t}{RC}$$

i.e $\ln Q$ and t are linearly related,
We can find $\ln Q$ values from given Q, then find the coefficients and intercepts of $\ln Q = mt + c$,
$m = -\frac{1}{RC}$ and $c = \ln(CV_0)$

$$\Rightarrow CV_0 = e^c$$

$$\Rightarrow C = \frac{e^c}{V_0}$$

Also,

$$RC = -\frac{1}{m}$$

$$\Rightarrow R = -\frac{1}{Cm} = -\frac{1}{\frac{e^c}{V_0}m} = -\frac{V_0}{e^c m}$$

Obtained Values

$R \approx 10^5$
$C \approx 5 \times 10^{-5}$