Effect on Bias

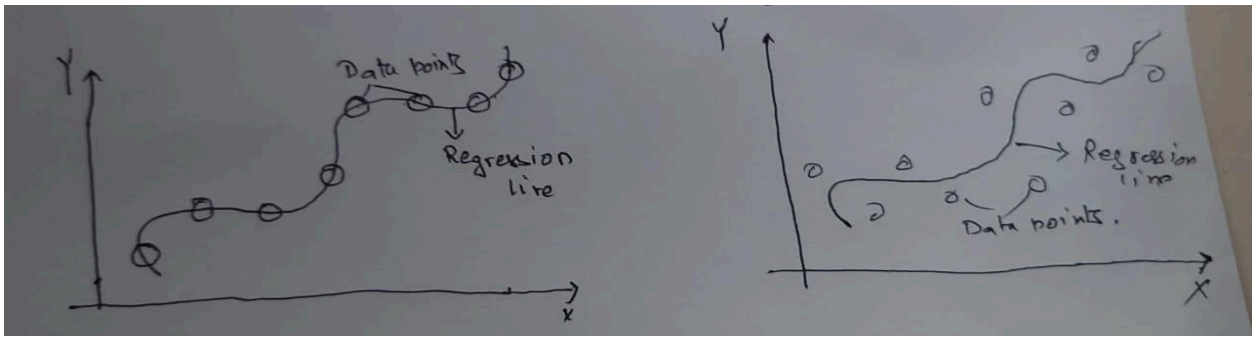
Bias Reduction: The bias of a model generally decreases as the model gets complex. Bias is introduced into the model because in real life, most problems may be complex, but we approximate the problem with a simplified model. More flexible, the model can learn with higher-order terms and additional features, complicated patterns in the data. These cut down on the simplifications and assumptions about the underlying data. The resultant model is well-suited to fit the training data better and, hence, reduces bias.

Risk of Overfitting: While increasing the complexity reduces bias, it raises the risk of overfitting. Overfitting means the model can learn from real patterns, but also from the noise and random fluctuations of the data set it was trained on. If overfitting happens, then a model can perform extremely well on a training dataset but extremely bad on any other data it hasn't seen yet-or held out for validation-because it fails to generalize. This is because the model might get too fitted into the particularities of the training data and will not generalize a trend.

Effect on Variance

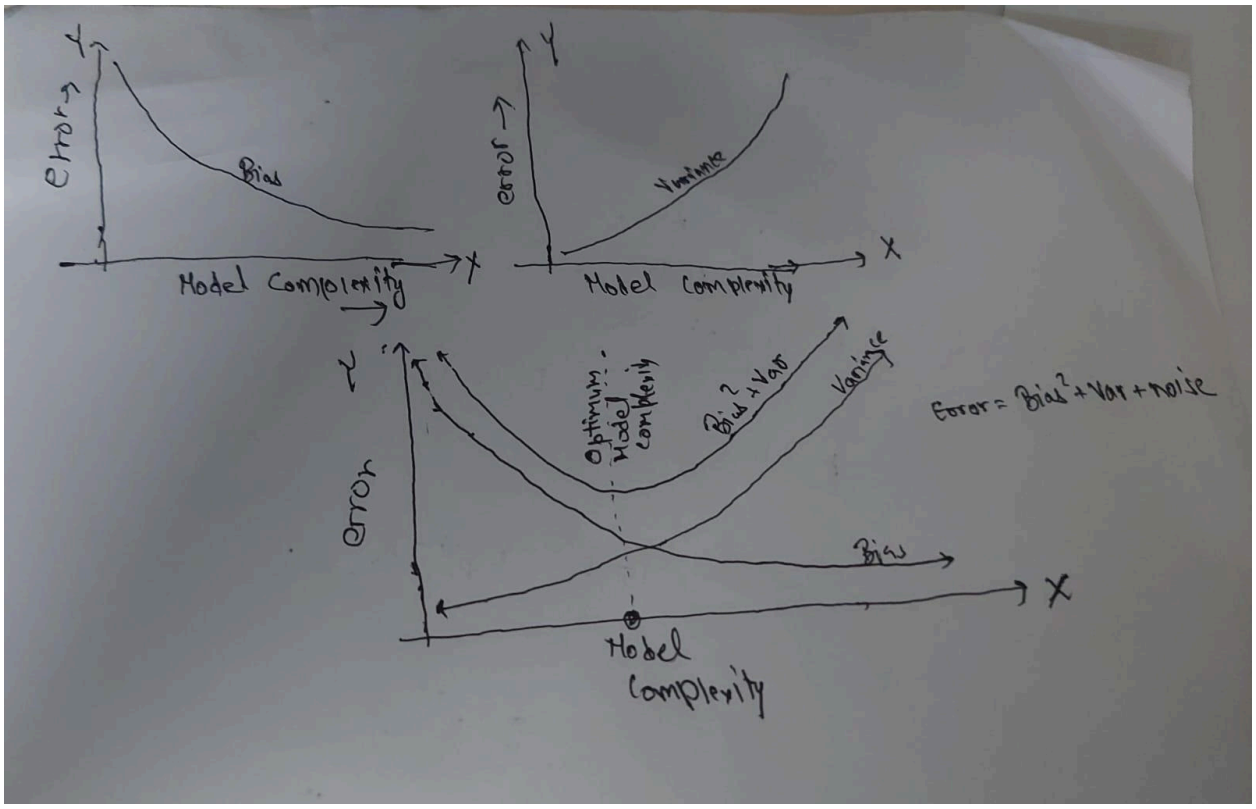
Increase in Variance: As we increase the complexity of a model its variance increases; this is because variance is a measure of how much its output varies with changes in the input of the training set. A complex model can perfectly fit any training dataset by making use of extra parameters or features it has. The implication here is that it will capture most of noise present in given data thereby making other observations difficult leading to higher level of unpredictability since particular information becomes increasingly dependent on individual sets on which they were developed or trained through.

Risk of Overfitting: During this process more instances there's a bigger chance for overfitting and eventual errors due to generalization difficulties encountered by models designed with high degrees such as multiple degree polynomial ones where merely adding one additional feature might drastically alter precision rate irrespective if there was any real difference whatsoever between them other than their size.



Reference used → Course Slides

In this example, the model overfits by capturing noise in the training data, resulting in high accuracy on training data but poor performance on testing data due to lack of generalization.

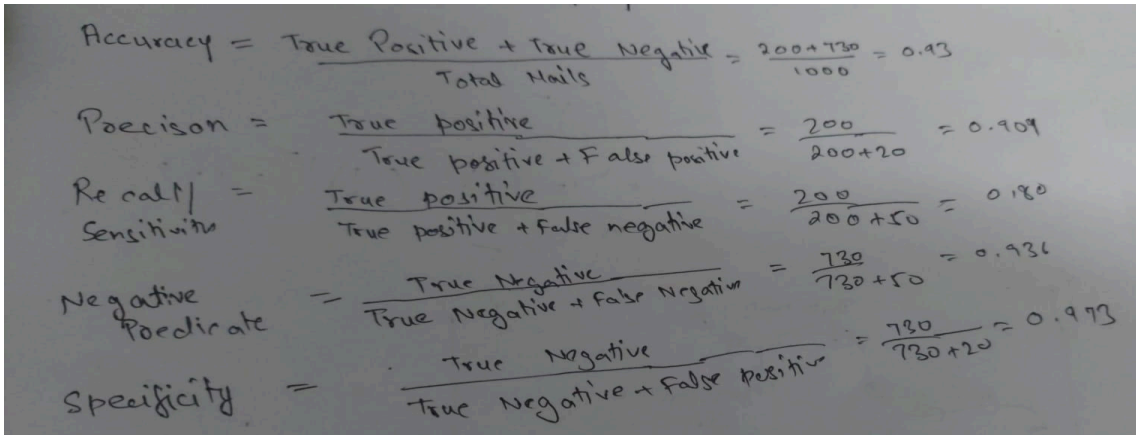


Reference used → Course Slides

Bias and variance are interdependent; as the bias increases, variance decreases, so we need to find an optimal point such that bias^2+ var is minimum so to minimize the total error.

Question 2>>

1. Spam Identified Correctly (True Positives for Spam): 200
2. Spam Classified as Not Spam (False Negatives): 50
3. Not Spam Identified Correctly (True Negatives): 730
4. Not spam Emails as Spam (False Positives): 20



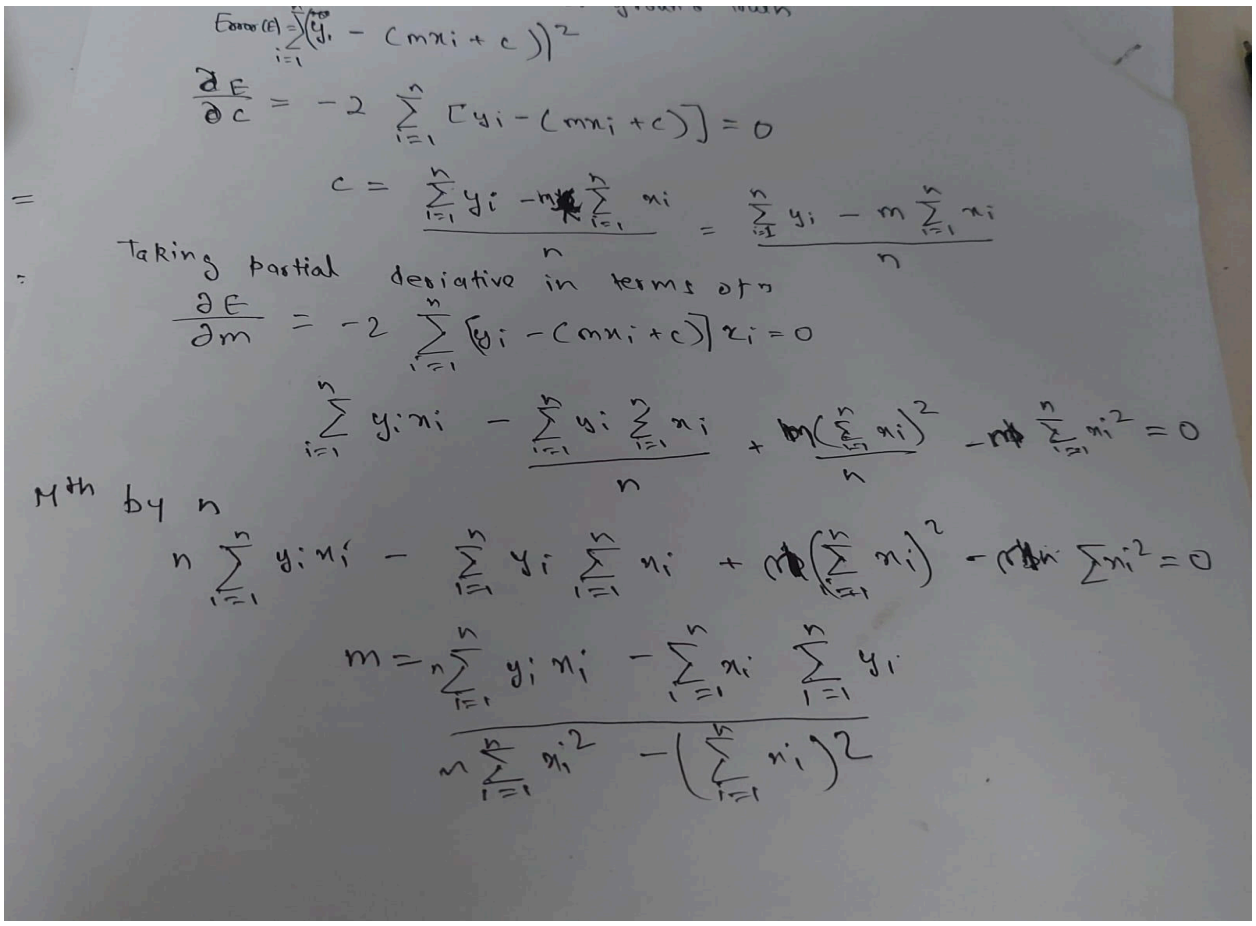
- Overall Accuracy: 93%
- Precision: 90.9%
- Recall/Sensitivity: 80%
- Negative Predicate: 93.6%
- Specificity:97.3%

Question 3>>

Let there be a linear regression model represented by

y=mx+c

to find m, we need to minimize the sum of squared residuals (errors). The residual for each data point (xi,yi):



For using Formula easily calculating m,c values

∑x=3+6+10+15+18=52

∑y=15+30+55+85+100=285

∑x2=32+62+102+152+182

∑x2=9+36+100+225+324=694

$\sum xy=(3\cdot 15)+(6\cdot 30)+(10\cdot 55)+(15\cdot 85)+(18\cdot 100)=45+180+550+1275+1800=3850$

$m= (5\cdot 3850-52\cdot 285) / (5\cdot 694-52\cdot 52)$

$m=4430/766$

$m=5.78$

$c=285-((5.78)\cdot 52) / 5$

$c =3.11$

equation of line: $y=mx+c$

so $y=5.78x+c$

Predicting the value for new point $x=12$

$y=5.78*12+3.11$

$y=66.25$

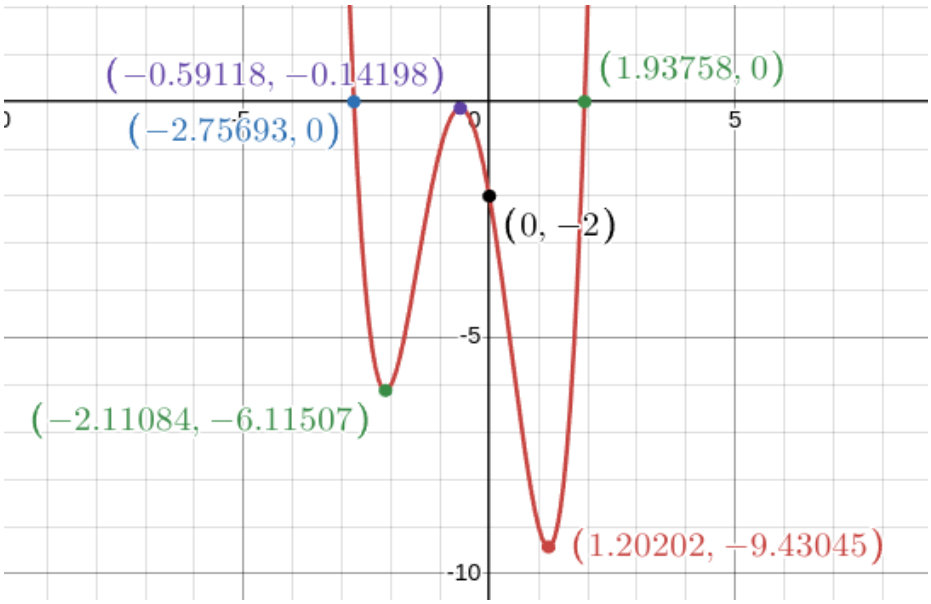
my predicted value for $x=12$ is 66.25

Question 4>>

f1 and f2, and the empirical risk for f1 is lower than that for f2, f1 has a lower empirical risk on the training set but may not necessarily generalize better than model f2 is a classical example of overfiiting

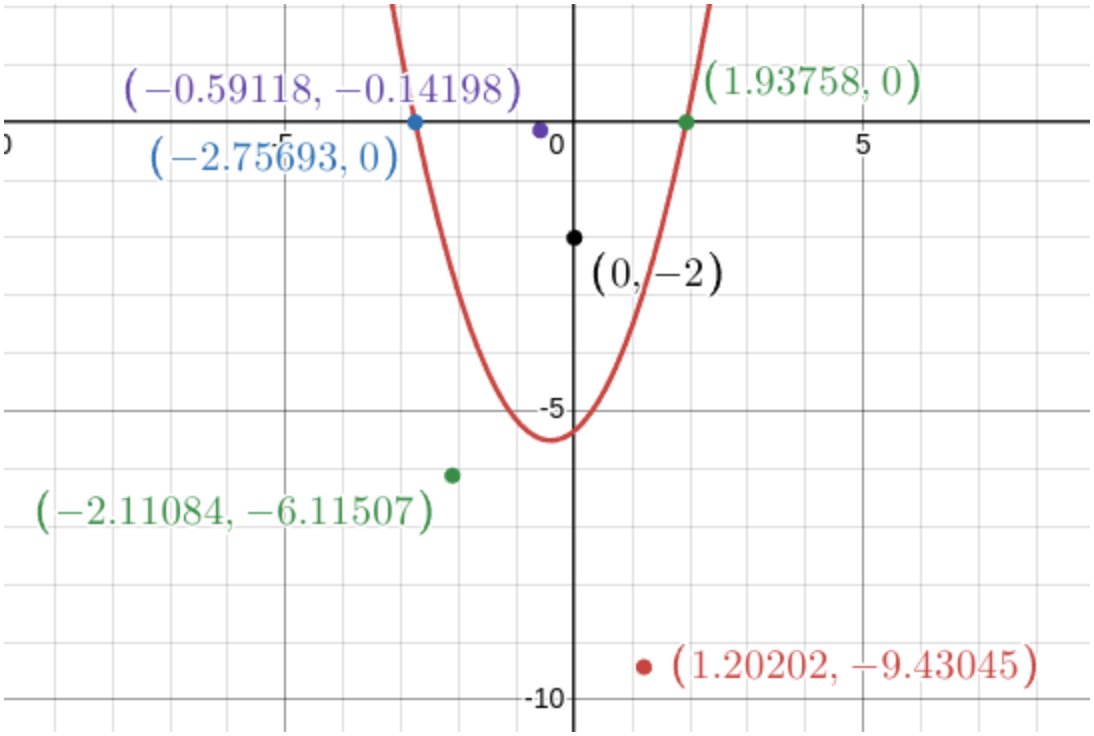
let the points be:

[(-0.59118,-0.14198),(0,-2),(-2.75693,0),(1.93758,0),(-2.11084,-6.11507),(1.20202,-9.43045)]



f1(x)=x**4 + 2*x**3 - 4*x**2 - 6*x - 2

mse(f1(x))=0

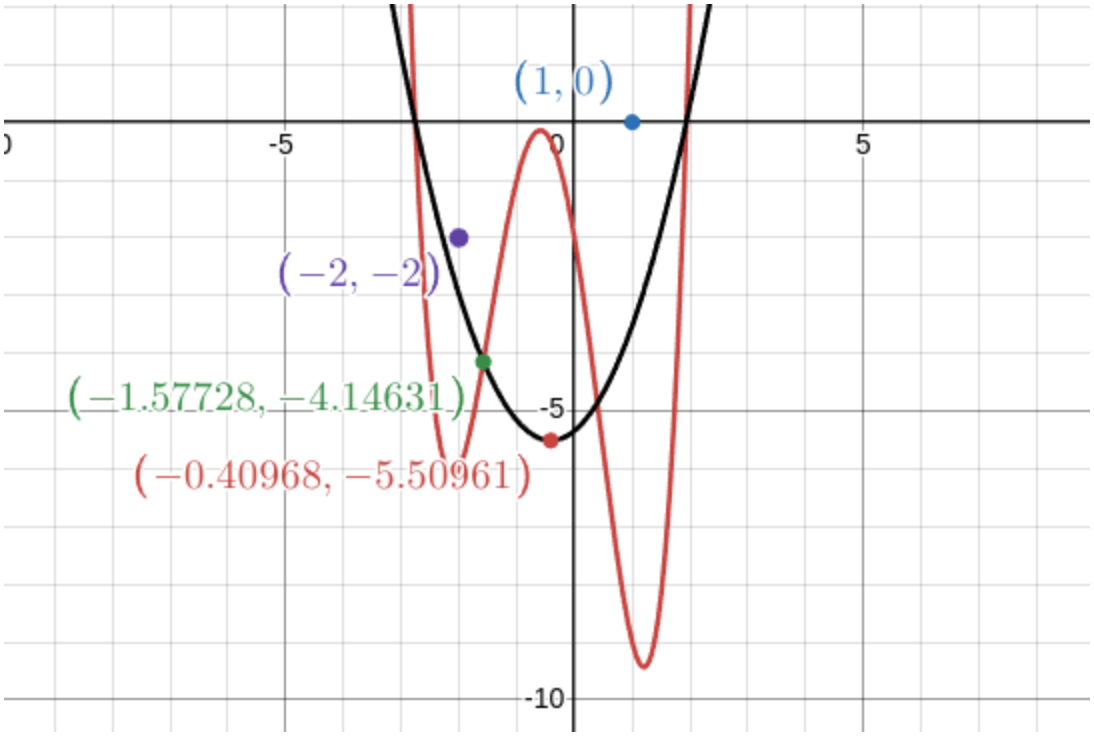


f2(x)=x**2 + 0.81935*x - 5.33554

Mean Squared Error of f2(x): 15.729802122259997

f1(x) empirical risk is less compared to f2(x), as we can see from the training set.

Let's see how it performs on the testing set.



let the testing points be [(-2,-2),(1,0),(-1.57728,-4.14631),(-0.40968,-5.50961)]

Mean Squared Error of f2(x) : 3.3282033742108026

Mean Squared Error of f1(x) : 30.97621420109041

Mean Squared Erroe

oh to my vision, f1(x) is performing poorer on f2(x).

so this is an classical example of overfitting here we say empirical risk in f1 is less in f2 and not generalizing better on f2(x) so

Note 1: Desmos used for graph generation and visualization
Note 2: Python code for mse calculation used

```
import numpy as np
points = [(-2,-2),(1,0),(-1.57728,-4.14631),(-0.40968,-5.50961)]
def model(x):
   return x**4 + 2*x**3 - 4*x**2 - 6*x - 2
squared_errors = []
for x, y in points:
   y_pred = model(x)
   squared_error = (y - y_pred) ** 2
   squared_errors.append(squared_error)
mse = np.mean(squared_errors)
print("Mean Squared Error:", mse)
```

Hypertune points and model accordingly