# Task 3

## 1. What is the difference between overfitting and underfitting?

Overfitting: the model performance is extremely good at the data set but when it see a new instances it would performance badly meaning it can not generalize to new instances We know that it overfits when the error on the validation set is extremely bigger than the Training set error

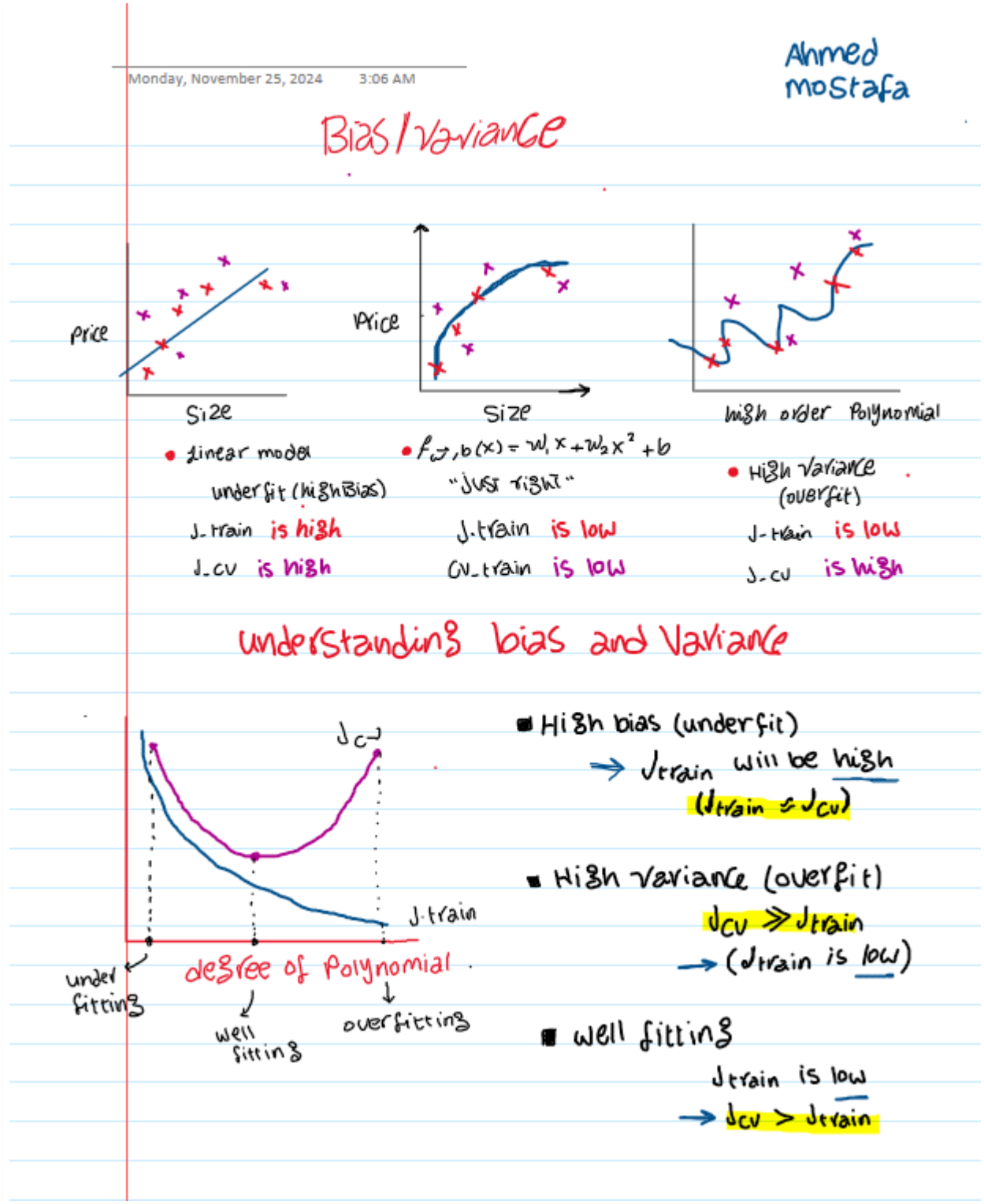
Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. **The possible solutions are:**

- To simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data or by constraining the model
- To gather more training data
- To reduce the noise in the training data (e.g., fix data errors and remove outliers)

Underfitting: the model performance is bad on either the training set and validation set Due to bad or weak algorithms like using linear regression on data that seems to be not linear, and we solve this problem by increase the polynomials of the algorithms

**The main options to fix this problem are**:

- Selecting a more powerful model, with more parameters
- Feeding better features to the learning algorithm (feature engineering)
- Reducing the constraints on the model (e.g., reducing the regularization hyper parameter)




Some of my study in Andrew NG Course

## 2. What do you know about ensemble learning?

Rather than using a single predictor (regressor or classifier) it's better to train multiple Predictors and get the aggregate of them and the result would be better than the best individual predictor so this group of predictors is called ensemble

This is done using some methods like:

Bootstrap(in tree ensembles) : if we have a training set of size m we use sampling with replacement to generate a new training sets with size m and train the trees on this training sets but the if we only do that the issue would be the trees would be almost identical so we fix it : when choosing a feature to split at each node if n features are available , we just pick a random subset of k<n features and only choose from this subset of features ***"and that's typically what happens when using random forest algorithm"***

Boosting: use sampling with replacement to create a new training set of size m, but instead of picking from all examples of equal probability, make it more likely to pick examples that the previous trained tree misclassify so we can see that the trees are trained sequentially ***"XGBoost"***

For prediction :

In classification it takes the most frequently class "the majority vote"

For regression it averages the predictions for all trees

## 3. What do you know about random forest?

Like I described in the previous question it is one of the learning ensemble algorithms that use tree encamping

***It uses one of the methods of ensemble learn which is bootstrap***

if we have a training set of size m we use sampling with replacement to generate a new training sets with size m and train the trees on this training sets but the if we only do that the issue would be the trees would be almost identical so we fix it : when choosing a feature to split at each node if n features are available , we just pick a random subset of k<n features and only choose from this subset of features ***"and that's typically what happens when using random forest algorithm***

## Tree ensembles

"using multiple decision tree"

The problem with the trees is that it's extremly sensitive to the small change

we overcome this problem by using multiple trees and pick the most promising one

## Sampling with replacment

- picking object from the set and return it to the set before picking another one
- by this criteria we can make multiple dataset "similar to the original one but not identical & an object can appear multiple time"

## Generating a tree sample

- Giving training set of size m
  use sampling with replacment to create a new training set of size m and train a decision tree on the new data set
  "the issue that the trees generated by that method is almost identical" Bagged decision tree"
  we can overcome this problem by
  "Randomizing the feature choice"
- At each node when choosing a feature to use to split, if n features are available, pick a random subset of $k < n$ features and allow the algorithm to only choose from the subset of features $K = \sqrt{n}$ "Random forest algorithm"

***These information I gained from Anrew NG course and that is a part from my study***

# What do you know about backpropagation?

Backpropagation, short for **backward propagation of errors**, is a fundamental algorithm in training neural networks. It helps the model optimize its weight by calculating the gradient of the loss function with respect to each weight in the network. This process ensures that the model learns to minimize the error during predictions.

### How It Works

1. **Forward Pass**:
    - The input data flows through the network layer by layer, and predictions are made.
    - The error (difference between predicted and actual values) is computed using a loss function.
2. **Backward Pass**:
    - The algorithm computes how much each weight in the network contributes to the error using the chain rule of calculus.
    - The gradients (partial derivatives of the loss function) are calculated for each weight.
3. **Weight Updates**:
    - The weights are updated using a technique like Gradient Descent

### Backpropagation is a good algorithm

Enables neural networks to learn complex patterns by adjusting weights systematically. Works well for deep networks with many layers, forming the basis of deep learning.

**One of the problems of it** Requires techniques like regularization to prevent the model from memorizing training data.

Backpropagation is at the core of modern machine learning and powers applications like image recognition, language modeling, and more.