.Bagging / Bootstrap Aggregation

Wednesday, April 19, 2023 10:09 AM

We regularly come across many game shows on television and you must have noticed an option of "Audience Poll". Most of the time a contestant goes with the option which has the highest vote from the audience and most of the time they win. We can generalize this in real life as well where taking opinions from a majority of people is much more preferred than the opinion of a single person. The Ensemble technique has a similar underlying idea where we aggregate predictions from a group of predictors, which may be classifiers or regressors, and most of the time the prediction is better than the one obtained using a single predictor.

Definition: — Ensemble learning is a machine learning paradigm where multiple models (often called "weak learners") are trained to solve the same problem and combined to get better results. The main hypothesis is that when weak models are correctly combined, we can obtain more accurate and/or robust models.

Weak Learners: A 'weak learner' is any ML algorithm (for regression/classification) that provides an accuracy slightly better than random guessing.

In ensemble learning theory, we call weak learners (or base models) models that can be used as building blocks for designing more complex models by combining several of them. Most of the time, these basics models perform not so well by themselves either because they have a high bias or because they have too much variance to be robust. Then, the idea of ensemble methods is to try reducing bias and/or variance of such weak learners by combining several of them together to create a strong learner (or ensemble model) that achieves better performances.

Let's suppose we have 'n' predictors/models:

Z1, Z2, Z3, ……., Zn with a standard deviation of σ

Variance(z) = σ^2

If we use single predictors Z1, Z2, Z3, ……., Zn the variance associated with each will be σ^2 but the expected value will be the average of all the predictors.

Let's consider the average of the predictors:

$\mu = (Z1 + Z2 + Z3 + \ldots + Zn)/n$

If we use μ as the predictor then the expected value remains the same but see the variance now:
variance(μ) = σ^2/n

So, the expected value remained 'μ' but variance decreases when we use an average of all the predictors.
This is why taking mean is preferred over using single predictors.

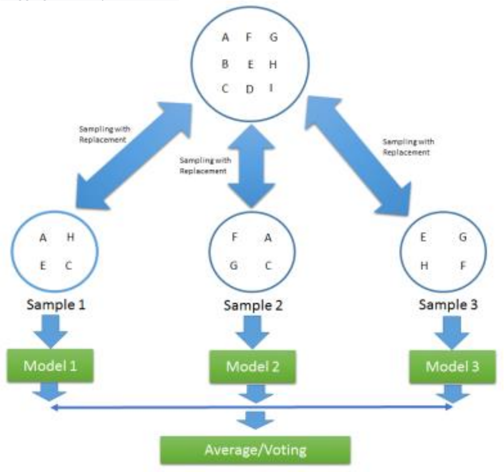
Ensemble methods take multiple small models and combine their predictions to obtain a more powerful predictive power.
There are few very popular Ensemble techniques which we will talk about in detail such as Bagging, Boosting, and stacking.

BAGGING

- Bagging stands for Bootstrap Aggregation.
- In real-life scenarios, we don't have multiple different training sets on which we can train our model separately and at the end combine their result. Here, bootstrapping comes into the picture.
- Bootstrapping is a technique of sampling different sets of data from a given training set by using replacement. After bootstrapping the training dataset, we train the model on all the different sets and aggregate the result. This technique is known as Bootstrap Aggregation or Bagging.

Definition: — Bagging is the type of ensemble technique in which a single training algorithm is used on different subsets of the training data where the subset sampling is done with replacement (bootstrap). Once the algorithm is trained on all the subset s, then bagging predicts by aggregating all the predictions made by the algorithm on different subsets.
· For aggregating the outputs of base learners, bagging uses majority voting (most frequent prediction among all predictions) for classification and averaging (mean of all the predictions) for regression.

Bagging visual representation:



Advantages of a Bagging Model:

1. Bagging significantly decreases the variance without increasing bias.
2. Bagging methods work so well because of diversity in the training data since the sampling is done by bootstrapping.
3. Also, if the training set is very huge, it can save computational time by training the model on a relatively smaller data set and still can increase the accuracy of the model.
4. Works well with small datasets as well.

· Disadvantages of a Bagging Model:

1. The main disadvantage of Bagging is that it improves the accuracy of the model at the expense of interpretability i.e., if a single tree was being used as the base model, then it would have a more attractive and easily interpretable diagram, but with the use of bagging this interpretability gets lost.

2. Another disadvantage of Bootstrap Aggregation is that during sampling, we cannot interpret which features are being selected i.e., there are chances that some features are never used, which may result in a loss of important information.

Out of Bag Evaluation: -In bagging, when different samples are collected, no sample contains all the data but a fraction of the original dataset. There might be some data that are never sampled at all. The remaining data which are not sampled are called out of bag instances.

Chat GPT

Bagging (Bootstrap Aggregation) is a technique used in ensemble learning, where multiple models are **trained on different subsets of the training data** to improve the performance of the final model.

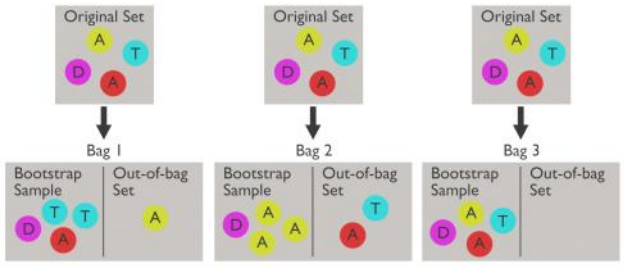
The bagging process involves the following steps:

1. **Random Sampling**: We randomly select m samples (with replacement) from the original dataset to create a new bootstrap sample. This process is repeated multiple times to create multiple bootstrap samples.

2. **Model Training:** A base model (such as a decision tree or a neural network) is trained on each bootstrap sample to create multiple models.

3. **Model Aggregation:** The predictions of all the models are combined (using methods like majority voting or averaging) to create the final prediction for each sample.

The aggregated predictions of the models are more robust and accurate than the predictions of any individual model, and this helps to improve the performance of the final model.
Overall, bagging helps to **reduce overfitting and improve the accuracy and stability** of the model.


Original Set
A
T
D
A
Bag 1
Bootstrap Sample
Out-of-bag Set
Bag 2
Bag 3

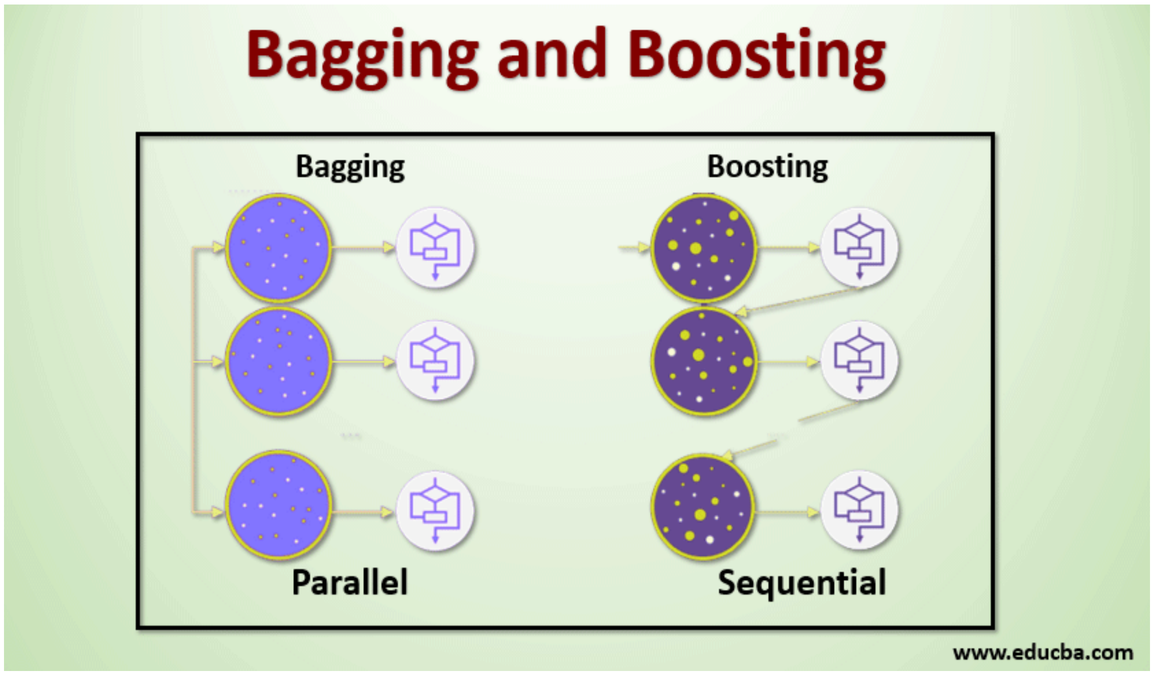
Boosting or Hypothesis Boosting

Wednesday, April 19, 2023 10:20 AM

Boosting, initially named Hypothesis Boosting, consists of the idea of filtering or weighting the data that is used to train our team of weak learners, so that each new learner gives more weight or is only trained with observations that have been poorly classified by the previous learners..

· By doing this our team of models learns to make accurate predictions on all kinds of data, not just on the most common or easy observations. Also, if one of the individual models is very bad at making predictions on some kind of observation, it does not matter, as the other N-1 models will most likely make up for it.

Definition: — The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor. The weak learners are sequentially corrected by their predecessors and, in the process, they are converted into strong learners.



· Also, in boosting, the data set is weighted (represented by the different sizes of the data points), so that observations that were incorrectly classified by classifier n are given more importance in the training of model n + 1, while in bagging the training samples are taken randomly from the whole population.

· While in bagging the weak learners are trained in parallel using randomness, in boosting the learners are trained sequentially, such that each subsequent learner aims to reduce the errors of the previous learners.

· Boosting, like bagging, can be used for regression as well as for classification problems.

· Boosting is mainly focused on reducing bias.

Any algorithm could have been used as a base for the boosting technique, but the reason for choosing trees are:

Pro's
· Computational scalability,
· Handles missing values,
· Robust to outliers,
· Does not require feature scaling,
· Can deal with irrelevant inputs,
· Interpretable (if small),
· Handles mixed predictors as well (quantitative and qualitative)

Con’s
· Inability to extract a linear combination of features
· High variance leading to a small computational power

And that’s where boosting comes into the picture. It minimizes the variance by taking into consideration the results from various trees.

· Advantages of a Boosting Model:
1. Boosting is a resilient method that curbs over-fitting easily.
2. Provably effective
3. Versatile — can be applied to a wide variety of problems.

· Disadvantages of a Boosting Model:
1. A disadvantage of boosting is that it is sensitive to outliers since every classifier is obliged to fix the errors in the predecessors. Thus, the method is too dependent on outliers.
2. Another disadvantage is that the method is almost impossible to scale up. This is because every estimator bases its correctness on the previous predictors, thus making the procedure difficult to streamline.

Ada Boost(Adaptive Boosting), Gradient Boosting, XG Boost(Xtreme Gradient Boosting) are few common examples of Boosting Techniques.

Stacking

Wednesday, April 19, 2023 10:41 AM

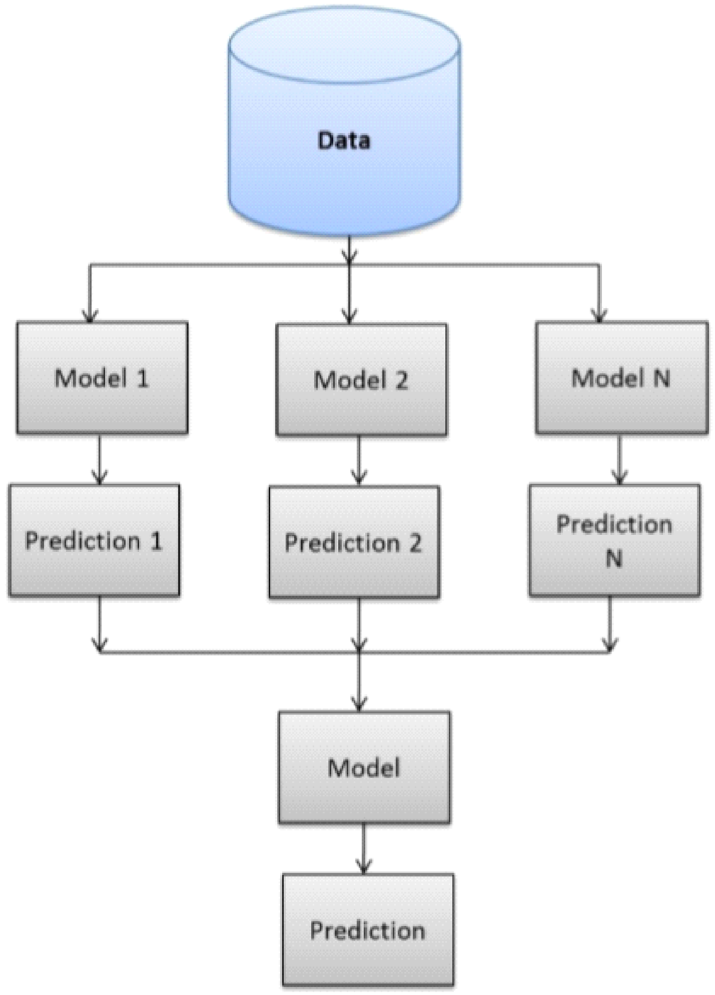
· Stacked Generalization or “Stacking” for short is an ensemble machine learning algorithm.

· Stacking mainly differs from bagging and boosting on two points. First stacking often considers heterogeneous weak learners (**different learning algorithms are combined**) whereas bagging and boosting consider mainly homogeneous weak learners.

Second, stacking learns to combine the base models using a meta-model whereas bagging and boosting combine weak learners following deterministic algorithms.

Definition: — Stacking is an ensemble learning method that combines multiple machine learning algorithms via meta-learning, In which base level algorithms are trained based on a **complete training data-set**, the meta-model is trained on the final outcomes of the all base-level model as a feature. We have a deal with bagging and boosting methods for handling bias and variance. Now we can learn stacking which is improve your model prediction accuracy.

Visual representation of Stacked Generalization :



In the above figure, we can see that different sample is not taken for training data to train classifiers. Instead of we are taking the whole data-set for training for every individual classifier. In this process, each classifier is working independently, which permits the classifiers with different hypotheses and algorithms. For Instance, **We can train our model on linear regression classifier, Decision Tree, and random forest for training, and then we can combine their prediction using a support vector machine.**

· Stacking, just like other ensemble techniques, tries to improve the accuracy of a model by using predictions of not so good models and then using those predictions as an input feature for a better model.

· Advantages of a Stacked Generalization Model:
1. The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a classification or regression task and make predictions that have **better performance** than any single model in the ensemble.

2. Stacking improves the model prediction **accuracy**.

· Disadvantage of a Stacked Generalization Model:
1. As we are taking the whole dataset for training for every individual classifier, in the case of huge datasets the **computational time** will be more as each classifier is working independently on the huge dataset.

Types of Boosting algorithms

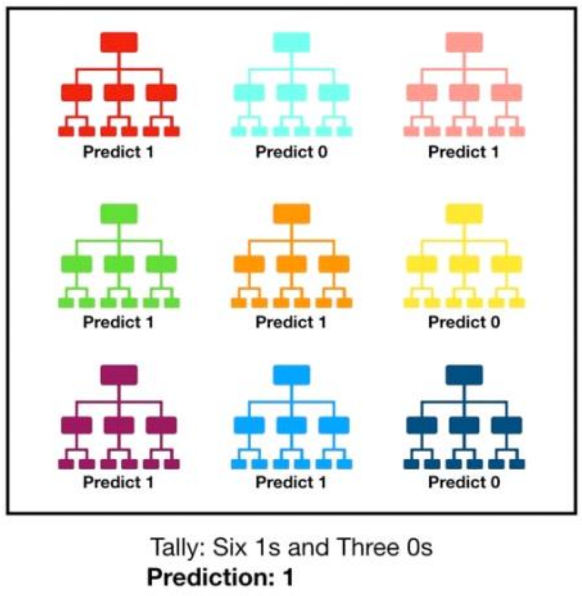
Wednesday, April 19, 2023 10:20 AM

Ada Boost(Adaptive Boosting), Gradient Boosting, XG Boost(Xtreme Gradient Boosting) are few common examples of Boosting Techniques.

Random Forest Classifier

Wednesday, April 19, 2023 10:23 AM

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).



Random forest allow each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.

For example, if our training data was [1, 2, 3, 4, 5, 6] then we might give one of our trees the following list [1, 2, 2, 3, 6, 6]. Notice that both lists are of length six and that "2" and "6" are both repeated in the randomly selected training data we give to our tree (because we sample with replacement).

Random forests are an ensemble learning method that builds multiple decision trees by randomly selecting subsets of the data and features for each tree. This randomness helps to reduce overfitting and improve the performance of the model.

During the training process, each decision tree is trained on a different random subset of the data and features, which makes the trees diverse and reduces their correlation(uncorrelated). This, in turn, leads to a more robust and accurate model that can generalize well to new, unseen data.

Therefore, random forests allow training on different sets of data and features, and this is actually a key feature of the algorithm that makes it effective.

Drawback of random Forest

1. **Interpretability:** Random forest models can be **difficult to interpret**, **especially if they involve a large number of trees and features**. It can be challenging to understand how the model is making its predictions and which features are most important.

2. **Overfitting:** While random forest is designed to reduce overfitting, it can still occur **if the trees in the forest are too deep or if the number of trees is too high.** Overfitting can result in a model that performs well on the training data but poorly on new, unseen data.

3. **Computational complexity:** Random forest can be computationally expensive to train and evaluate, **especially for large datasets with many features.** The algorithm involves **building multiple decision trees and selecting the best features for each tree,** which can be time-consuming.

4. **Imbalanced Data:** Random Forest has been shown to perform poorly on imbalanced datasets, where the number of examples in each class is significantly different. It may **give high accuracy on majority class but not as good performance on minority**

class.

5. **Hyperparameter Tuning:** Random Forest has a number of hyperparameters such as **the number of trees, the depth of the trees, the number of features to consider at each split,** etc. Tuning these hyperparameters can be challenging and time-consuming.

Gradient Descent

Wednesday, April 19, 2023 10:44 AM

Understanding the Mathematics behind Gradient Descent. | by Parul Pandey | Towards Data Science
Basic concept understanding in lay man language

Definition:

Gradient descent is an optimization algorithm that works iteratively to find the model parameters/coefficient(m, b of y=mx+b) with minimal cost or error values.

If we go through a formal definition: Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function.

Scenario to understand the Gradient descent concept:

you are playing a game with your friends where you have to throw a paper ball in a basket. What will be your approach to this problem?

1. Here, I have a ball and basket at a particular distance, If I throw the ball with 30% of my strength, I believe it should end up in the basket. But with this force ball didn't reach the target and falls before the basket.
2. In my second attempt, I will use more power say 50% of my strength. This time I am sure it will reach its place. But unexpectedly ball crosses the basket and falls far from it.
3. Now I calculate and conclude the force should be somewhere and between 30% and 50%. This time I threw the ball with 45% of my strength and yay! this time I succeed. The ball directly drops in the basket. That means 45% of strength is the optimum solution for this problem.
   This is how the gradient descent algorithm works.

Summary:

For a given problem statement, the solution starts with a Random initialization. These initial parameters are then used to generate the predictions i.e. the output. Once we have the predicted values we can calculate the error or the cost. i.e. how far the predicted values are from the actual target. In the next step, we update the parameters accordingly and again made the predictions with updated parameters.

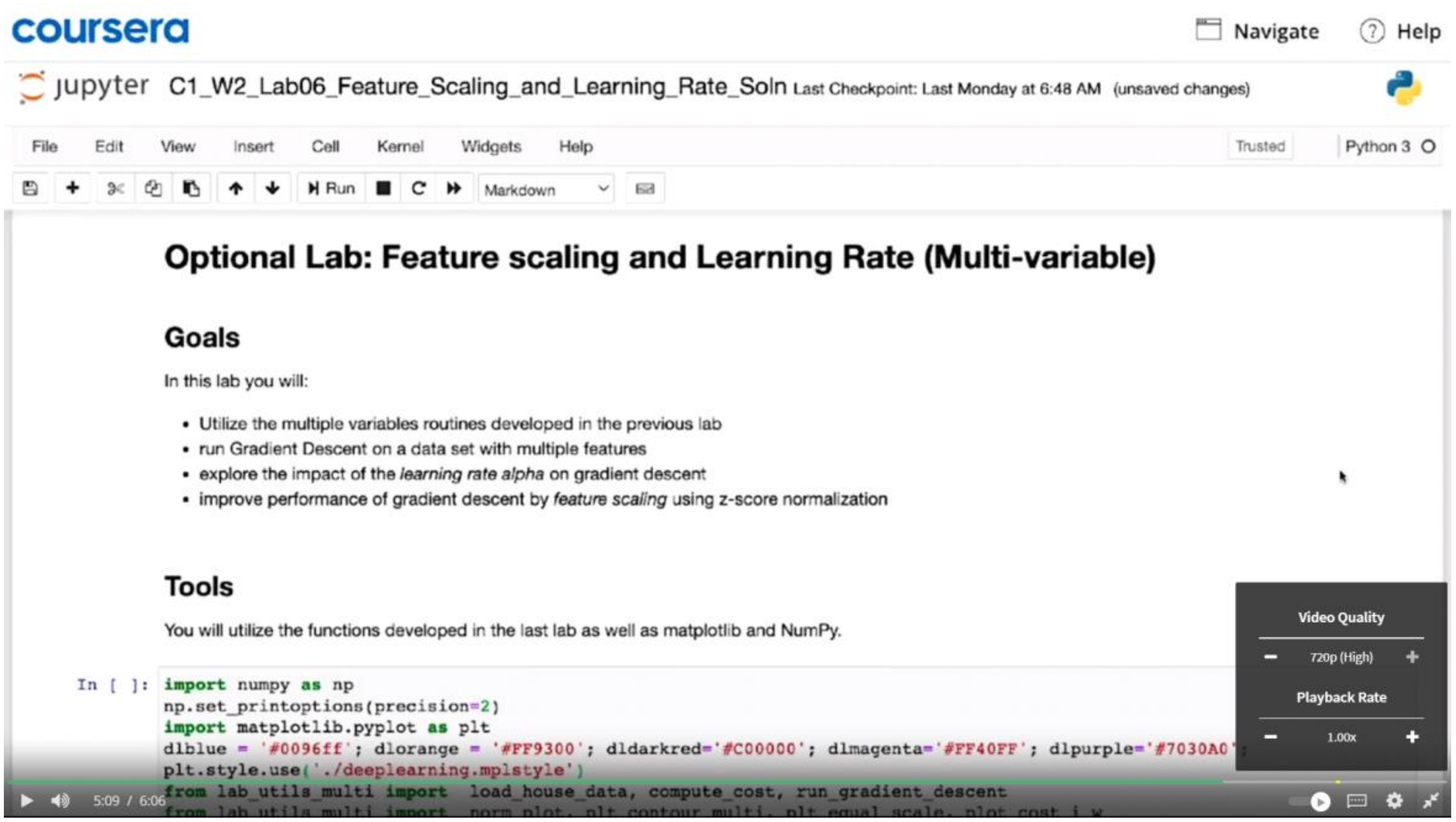
Popular variants of gradient descent:

1. **Batch Gradient Descent**: In this variant, the gradient is computed using the **entire training dataset**, and the parameters are **updated once per epoch**. This approach can be **slow** and computationally expensive, especially for large datasets.
2. **Stochastic Gradient Descent (SGD)**: In this variant, the gradient is computed using a randomly selected subset of the training data (i.e., a **batch**), and the parameters are **updated after each batch**. This approach is faster than batch gradient descent, but can be **noisy** and may require more iterations to converge.
3. **Mini-batch Gradient Descent**: This variant is a compromise between batch gradient descent and SGD, where the gradient is computed using a small randomly selected subset of the training data (i.e., a **mini-batch**), and the parameters are **updated after each mini-batch**. This approach can be **faster** than batch gradient descent and less noisy than SGD.
4. **Momentum Gradient Descent**: This variant introduces a momentum term that helps accelerate the convergence by **adding** a fraction of the **previous update** vector to the **current** update vector. This approach can help **overcome plateaus and local minima.**
5. **Adaptive Gradient Descent**: These variants use adaptive learning rates that adjust the step size of the gradient descent based on the **history** of the gradients. Examples include AdaGrad, RMSprop, and Adam. These approaches can **converge faster** and more reliably than fixed learning rate methods.
   These methods are particularly useful in problems with sparse gradients or in **deep learning** where the gradients may vary significantly across different layers.
   a. **Adagrad**: In this variant, the learning rate is adaptively scaled for each parameter based on the **historical gradient information**. This allows for larger updates for infrequent parameters and smaller updates for frequent parameters.
   b. **RMSprop**: In this variant, the learning rate is adaptively scaled for each parameter based on the **moving average of the squared gradient.** This helps the algorithm to converge faster in the presence of noisy gradients.
   c. **Adam**: In this variant, the learning rate is adaptively scaled for each parameter based on the moving average of the gradient and the squared gradient. This **combines** the benefits of **Momentum-based Gradient Descent, Adagrad, and RMSprop,** and is one of the most popular optimization algorithms for deep learning.
6. **Conjugate Gradient**: In this variant, the gradient is replaced by a search direction, which is a **conjugate vector of the previous search direction**. This approach can converge faster than other gradient descent variants when the objective function is quadratic or approximately quadratic.
   This can be more efficient in solving large-scale optimization problems with a symmetric and positive definite matrix. CGD requires less memory and fewer iterations than other GD methods and can converge faster.

Disadvantages of gradient descent and its variants:

- **Choice of learning rate**: The choice of learning rate is crucial for the convergence of gradient descent and its variants. Choosing a learning rate that is **too large** can lead to **oscillations** or **overshooting**, while choosing a learning rate that is **too small** can lead to **slow convergence** or getting **stuck in local minima**.
- **Sensitivity to initialization**: Gradient descent and its variants can be sensitive to the initialization of the model's parameters, which can affect the convergence and the quality of the solution.
- **Time-consuming**: Gradient descent and its variants can be time-consuming, especially when dealing with large datasets and high-dimensional models. The convergence speed can also vary depending on the variant used and the specific problem.
- **Local optima**: Gradient descent and its variants can converge to a local minimum instead of the global minimum of the cost function, especially in non-convex problems. This can affect the quality of the solution, and techniques like random initialization and multiple restarts may be used to mitigate this issue.

Implementation detail


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
Video Quality
720p (High)
Playback Rate
1.00x
5:09 / 6:06

Optional Lab: Feature scaling and Learning Rate (Multi-variable)

Goals

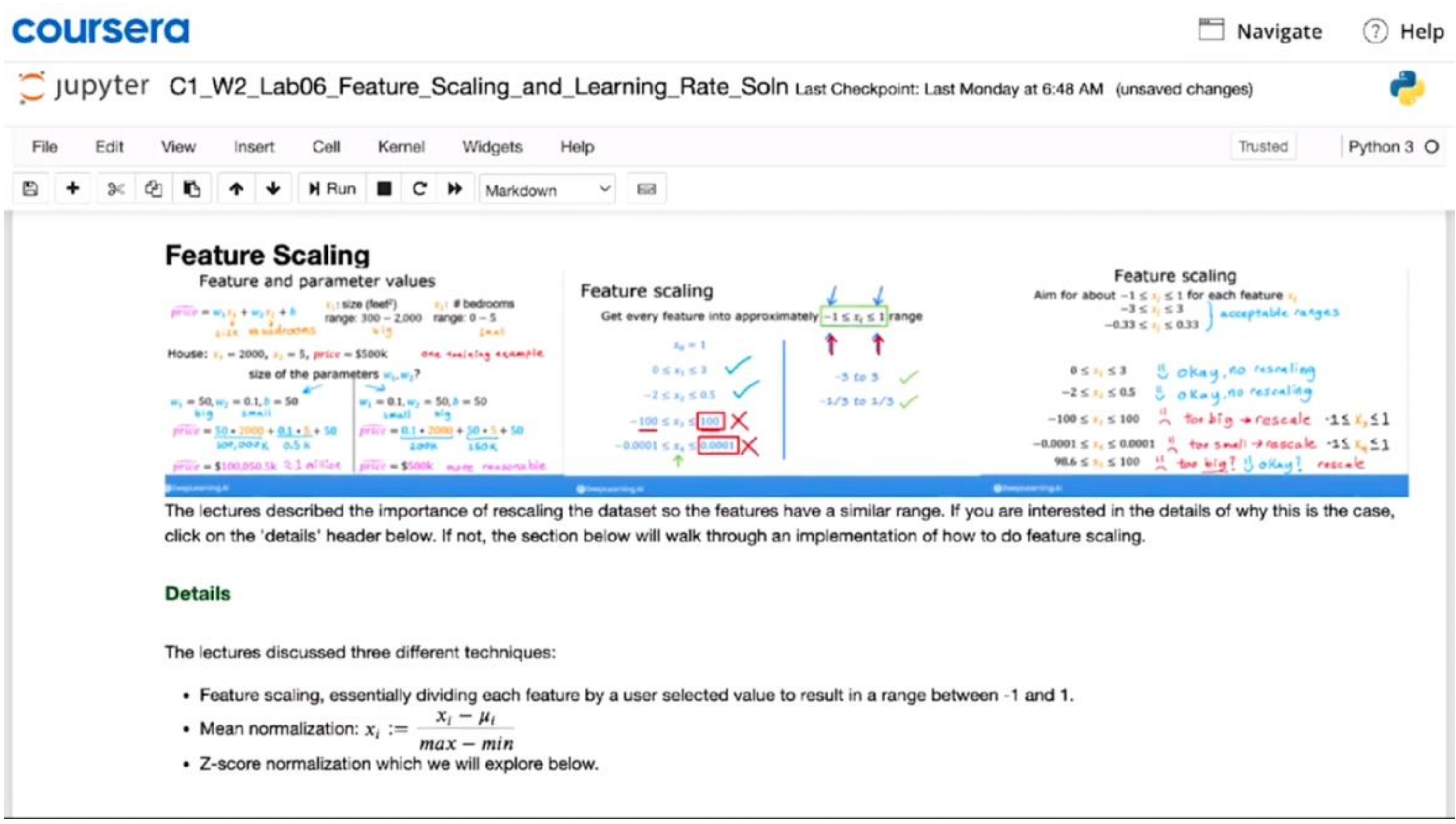
In this lab you will:

- Utilize the multiple variables routines developed in the previous lab
- run Gradient Descent on a data set with multiple features
- explore the impact of the *learning rate alpha* on gradient descent
- improve performance of gradient descent by *feature scaling* using z-score normalization

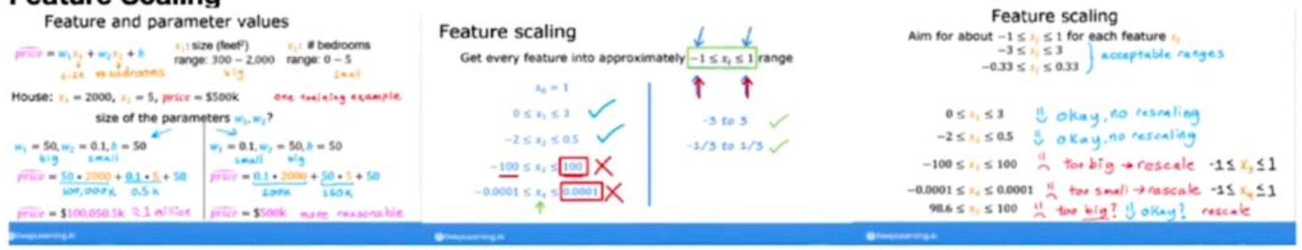
Tools

You will utilize the functions developed in the last lab as well as matplotlib and NumPy.

```
In [ ]: import numpy as np
        np.set_printoptions(precision=2)
        import matplotlib.pyplot as plt
        dlblue = '#0096ff'; dlorange = '#FF9300'; dldarkred='#C00000'; dlmagenta='#FF40FF'; dlpurple='#7030A0'
        plt.style.use('./deeplearning.mplstyle')
        from lab_utils_multi import  load_house_data, compute_cost, run_gradient_descent
        from lab_utils_multi import  norm_plot, plt_contour_multi, plt_equal_scale, plot_cost_i_w
```


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Feature Scaling

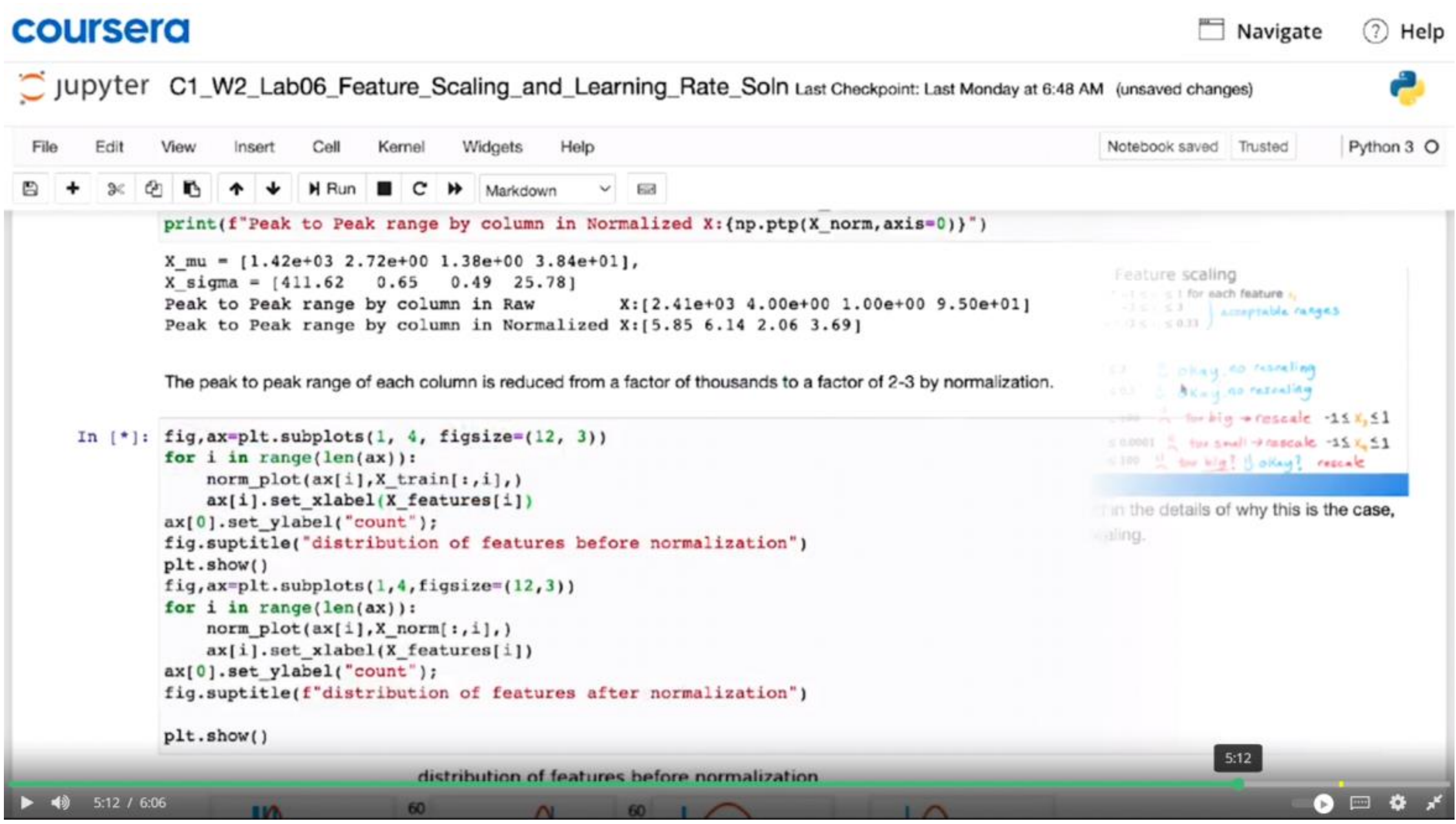

Feature and parameter values
x_1: size (feet²) range: 300 – 2,000 big
x_2: # bedrooms range: 0 – 5 small
House: x_1 = 2000, x_2 = 5, price = $500k
one training example
size of the parameters w_1, w_2?
w_1 = 50, w_2 = 0.1, b = 50
w_1 = 0.1, w_2 = 50, b = 50
price = $100,050.5k
price = $500k
more reasonable
Feature scaling
Get every feature into approximately $-1 \le x_j \le 1$ range
$x_0 = 1$
$0 \le x_1 \le 3$
$-2 \le x_2 \le 0.5$
$-100 \le x_3 \le 100$
$-0.0001 \le x_4 \le 0.0001$
-3 to 3
-1/3 to 1/3
Feature scaling
Aim for about $-1 \le x_j \le 1$ for each feature x_j
$-3 \le x_j \le 3$
$-0.33 \le x_j \le 0.33$
acceptable ranges
okay, no rescaling
too big → rescale
too small → rescale
too big? okay? rescale

The lectures described the importance of rescaling the dataset so the features have a similar range. If you are interested in the details of why this is the case, click on the 'details' header below. If not, the section below will walk through an implementation of how to do feature scaling.

Details

The lectures discussed three different techniques:

- Feature scaling, essentially dividing each feature by a user selected value to result in a range between -1 and 1.
- Mean normalization: $x_i := \dfrac{x_i - \mu_i}{max - min}$
- Z-score normalization which we will explore below.



```
print(f"Peak to Peak range by column in Normalized X:{np.ptp(X_norm,axis=0)}")
```

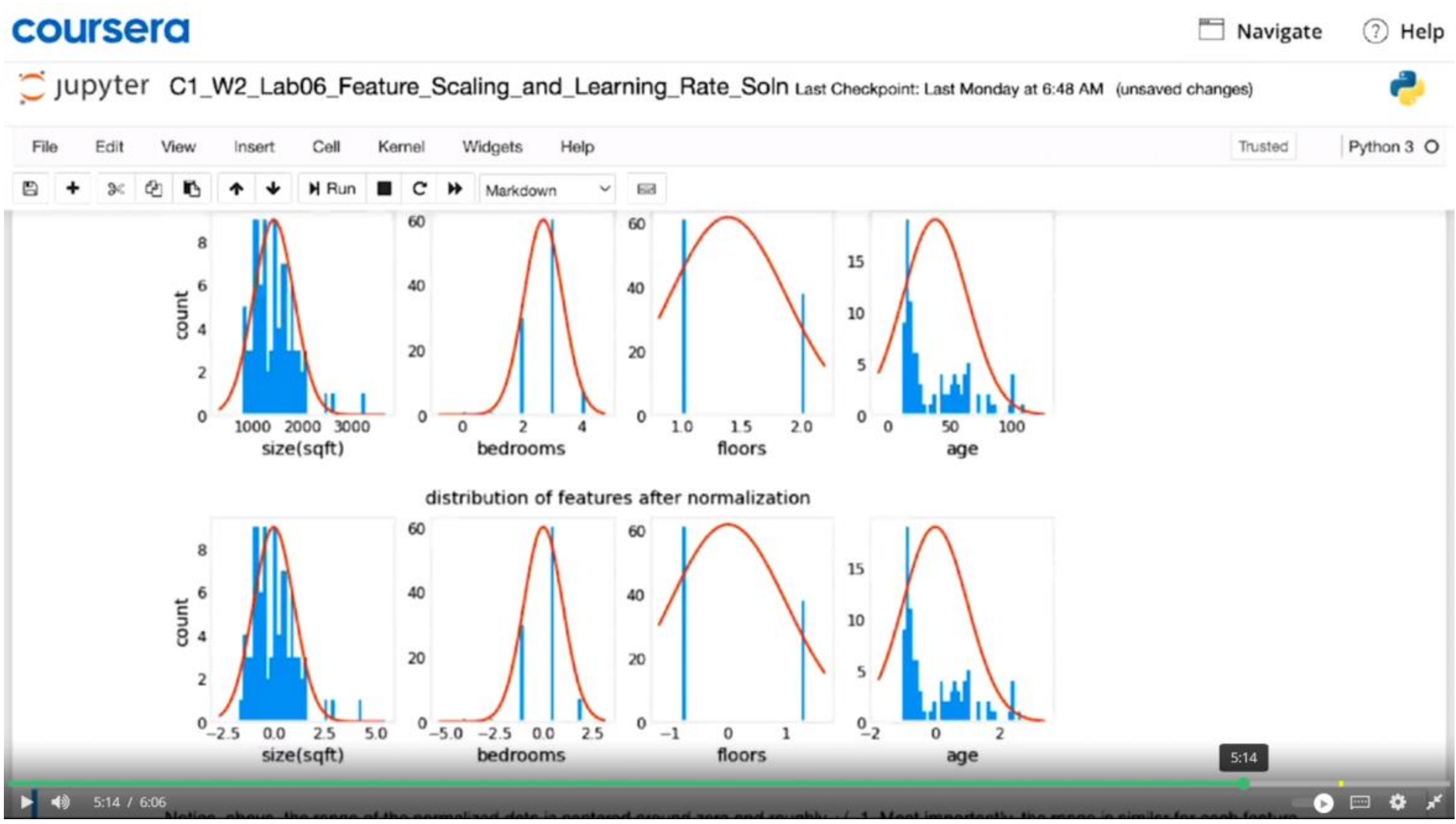
```
X_mu = [1.42e+03 2.72e+00 1.38e+00 3.84e+01],
X_sigma = [411.62   0.65   0.49  25.78]
Peak to Peak range by column in Raw        X:[2.41e+03 4.00e+00 1.00e+00 9.50e+01]
Peak to Peak range by column in Normalized X:[5.85 6.14 2.06 3.69]
```

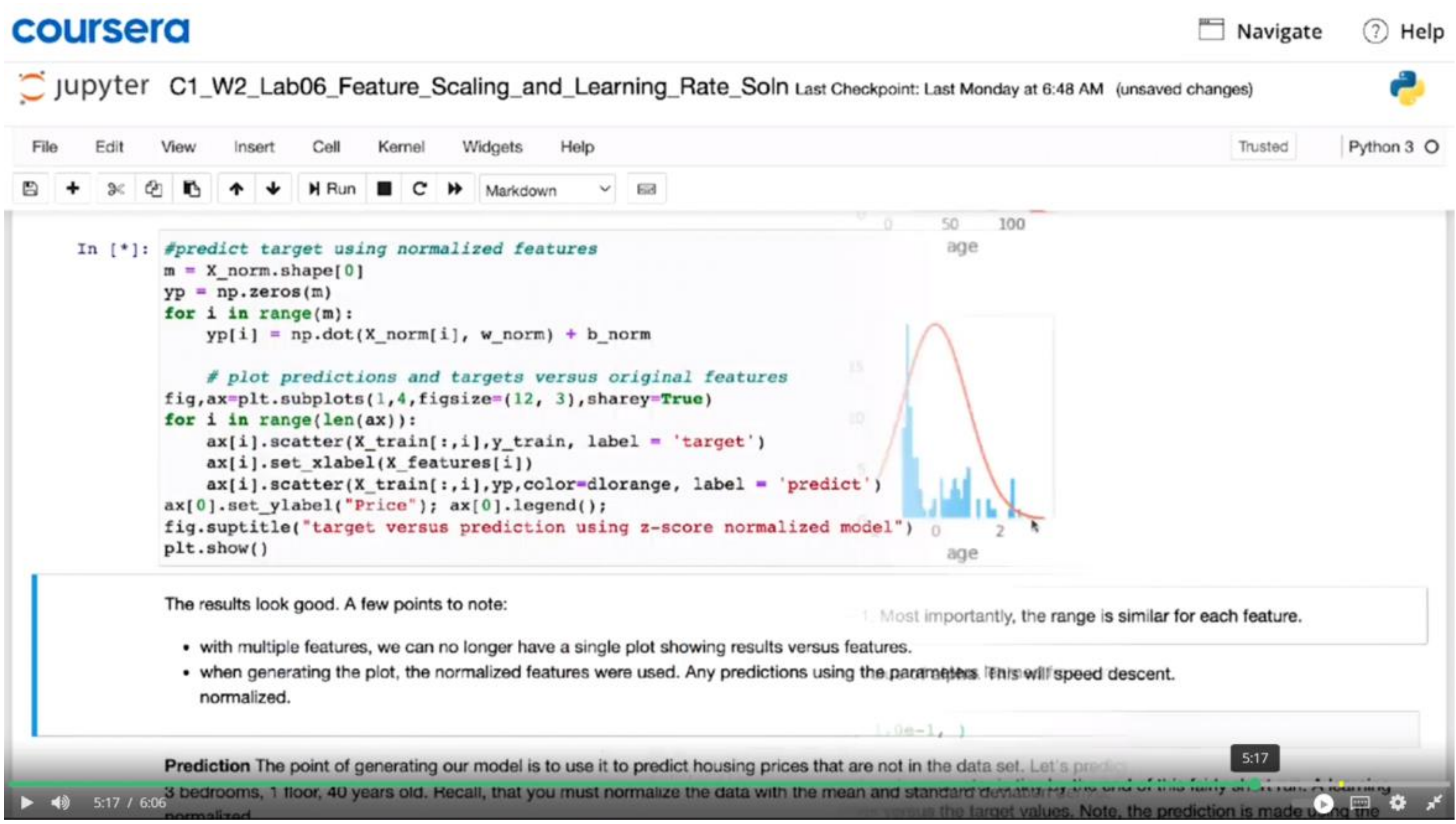
The peak to peak range of each column is reduced from a factor of thousands to a factor of 2-3 by normalization.

```
In [*]: fig,ax=plt.subplots(1, 4, figsize=(12, 3))
        for i in range(len(ax)):
            norm_plot(ax[i],X_train[:,i],)
            ax[i].set_xlabel(X_features[i])
        ax[0].set_ylabel("count");
        fig.suptitle("distribution of features before normalization")
        plt.show()
        fig,ax=plt.subplots(1,4,figsize=(12,3))
        for i in range(len(ax)):
            norm_plot(ax[i],X_norm[:,i],)
            ax[i].set_xlabel(X_features[i])
        ax[0].set_ylabel("count");
        fig.suptitle(f"distribution of features after normalization")

        plt.show()
```

in the details of why this is the case,

distribution of features before normalization


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
count
size(sqft)
bedrooms
floors
age
distribution of features after normalization
count
size(sqft)
bedrooms
floors
age
5:14
5:14 / 6:06


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
age
age
5:17 / 6:06
5:17

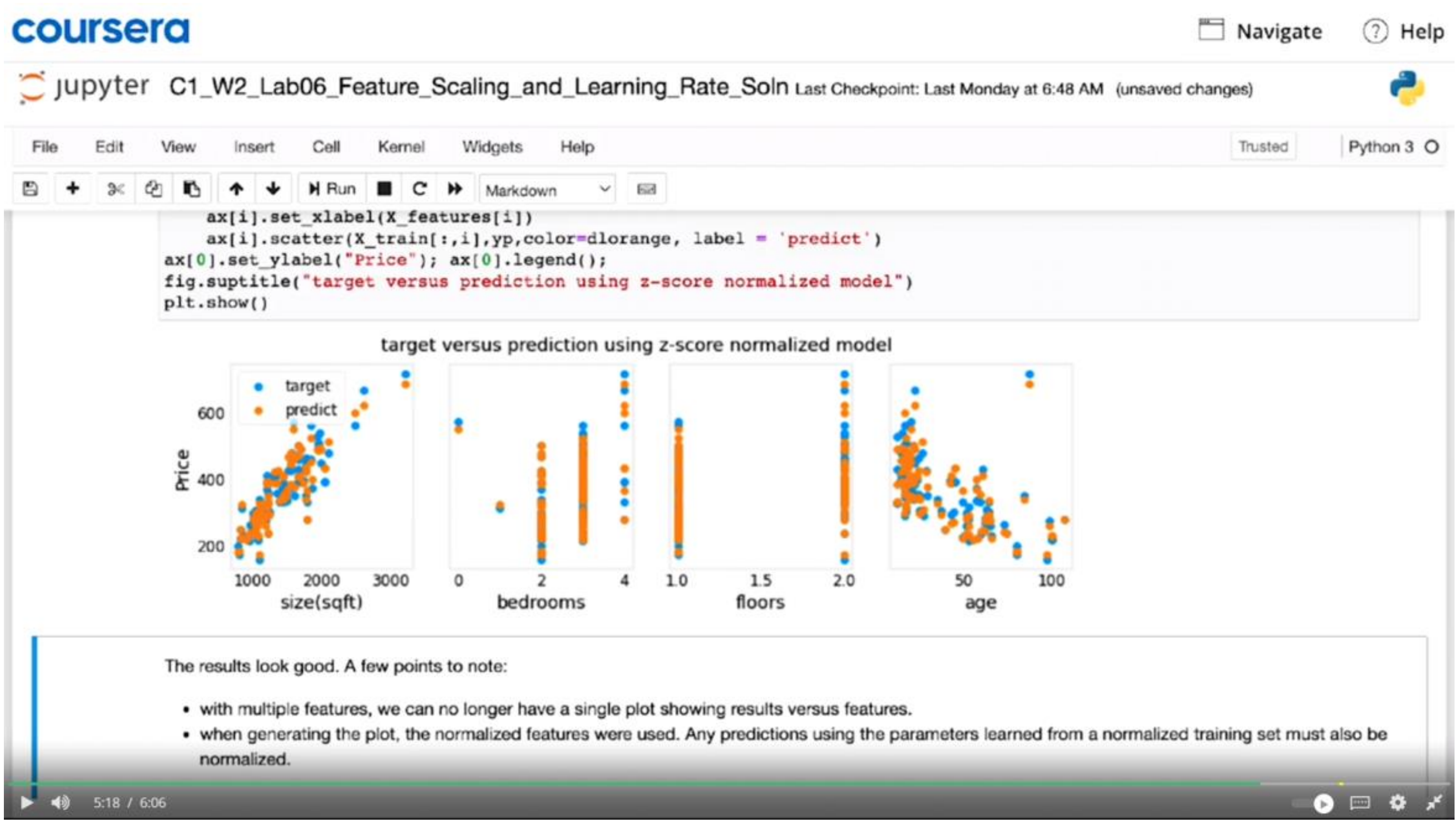
```
In [*]: #predict target using normalized features
        m = X_norm.shape[0]
        yp = np.zeros(m)
        for i in range(m):
            yp[i] = np.dot(X_norm[i], w_norm) + b_norm

            # plot predictions and targets versus original features
        fig,ax=plt.subplots(1,4,figsize=(12, 3),sharey=True)
        for i in range(len(ax)):
            ax[i].scatter(X_train[:,i],y_train, label = 'target')
            ax[i].set_xlabel(X_features[i])
            ax[i].scatter(X_train[:,i],yp,color=dlorange, label = 'predict')
        ax[0].set_ylabel("Price"); ax[0].legend();
        fig.suptitle("target versus prediction using z-score normalized model")
        plt.show()
```

1. Most importantly, the range is similar for each feature.

The results look good. A few points to note:

- with multiple features, we can no longer have a single plot showing results versus features.
- when generating the plot, the normalized features were used. Any predictions using the parameters This will speed descent. normalized.

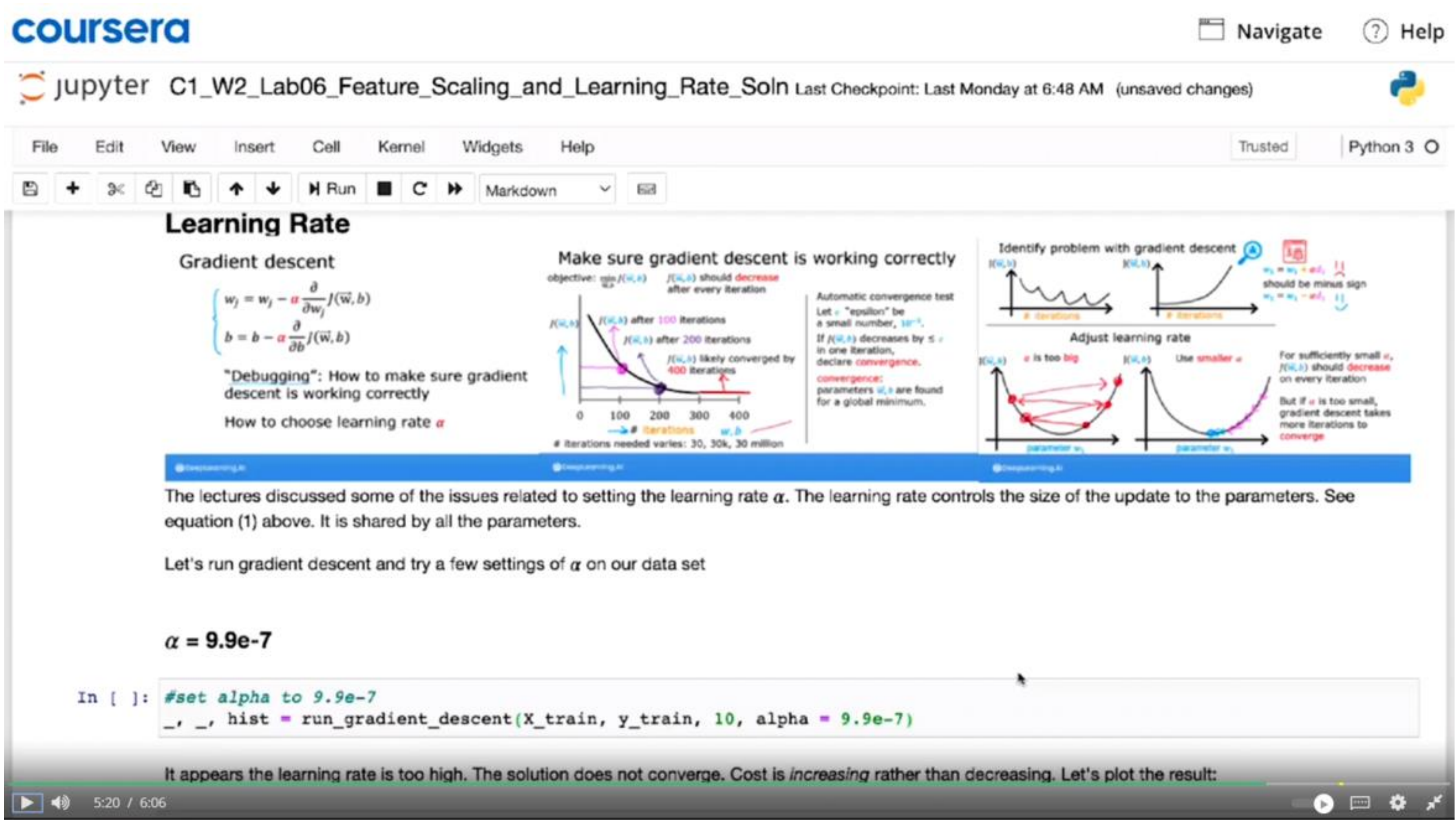
Prediction The point of generating our model is to use it to predict housing prices that are not in the data set. Let's predict 3 bedrooms, 1 floor, 40 years old. Recall, that you must normalize the data with the mean and standard deviation normalized ... versus the target values. Note, the prediction is made using the


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
target versus prediction using z-score normalized model
target
predict
Price
600
400
200
1000
2000
3000
size(sqft)
0
2
4
bedrooms
1.0
1.5
2.0
floors
50
100
age
5:18 / 6:06

```
            ax[i].set_xlabel(X_features[i])
            ax[i].scatter(X_train[:,i],yp,color=dlorange, label = 'predict')
        ax[0].set_ylabel("Price"); ax[0].legend();
        fig.suptitle("target versus prediction using z-score normalized model")
        plt.show()
```

The results look good. A few points to note:

- with multiple features, we can no longer have a single plot showing results versus features.
- when generating the plot, the normalized features were used. Any predictions using the parameters learned from a normalized training set must also be normalized.


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
Gradient descent
"Debugging": How to make sure gradient descent is working correctly
How to choose learning rate α
Make sure gradient descent is working correctly
Identify problem with gradient descent
Adjust learning rate
5:20 / 6:06

Learning Rate

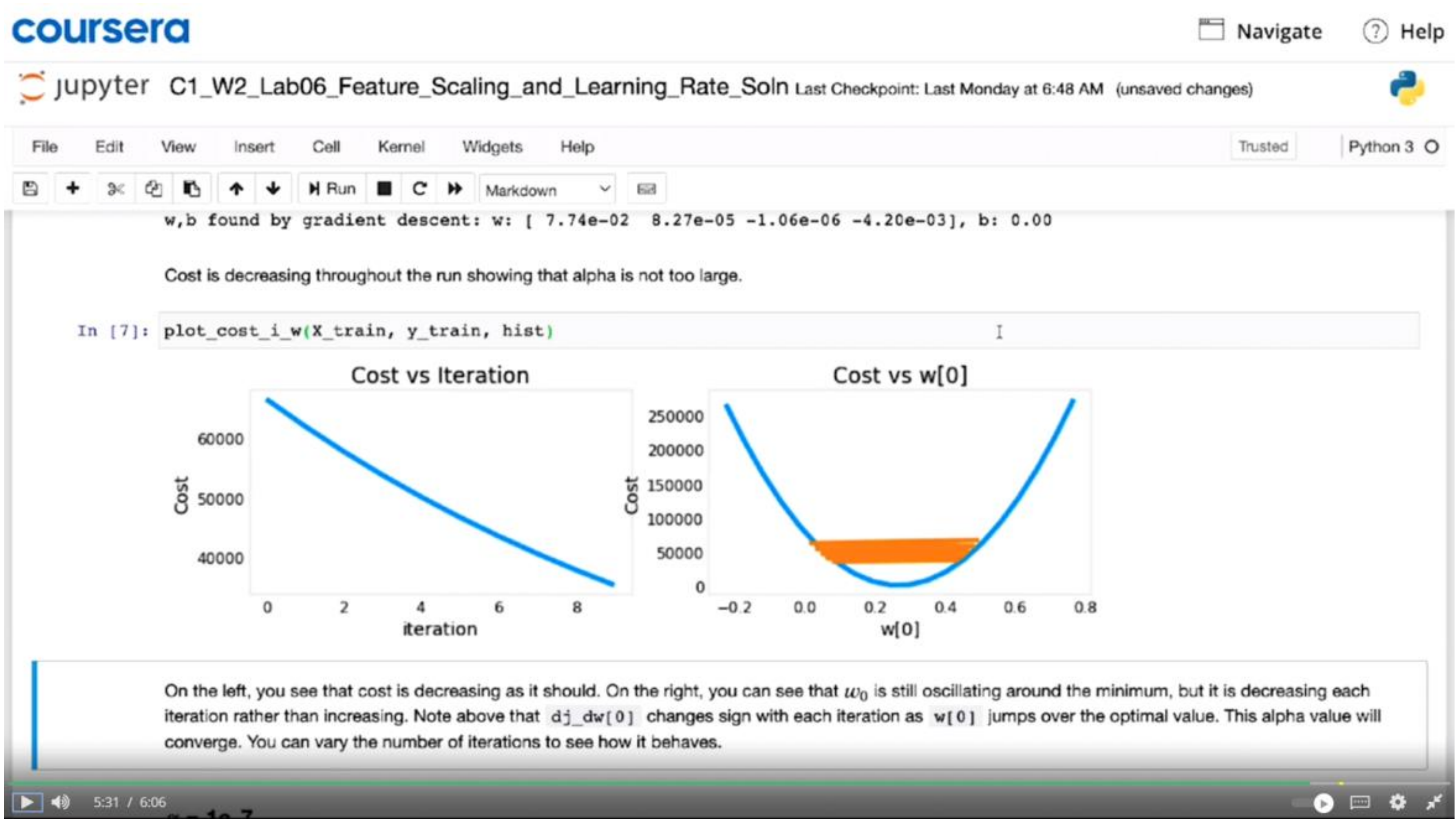
The lectures discussed some of the issues related to setting the learning rate α. The learning rate controls the size of the update to the parameters. See equation (1) above. It is shared by all the parameters.

Let's run gradient descent and try a few settings of α on our data set

α = 9.9e-7

```
In [ ]: #set alpha to 9.9e-7
        _, _, hist = run_gradient_descent(X_train, y_train, 10, alpha = 9.9e-7)
```

It appears the learning rate is too high. The solution does not converge. Cost is *increasing* rather than decreasing. Let's plot the result:

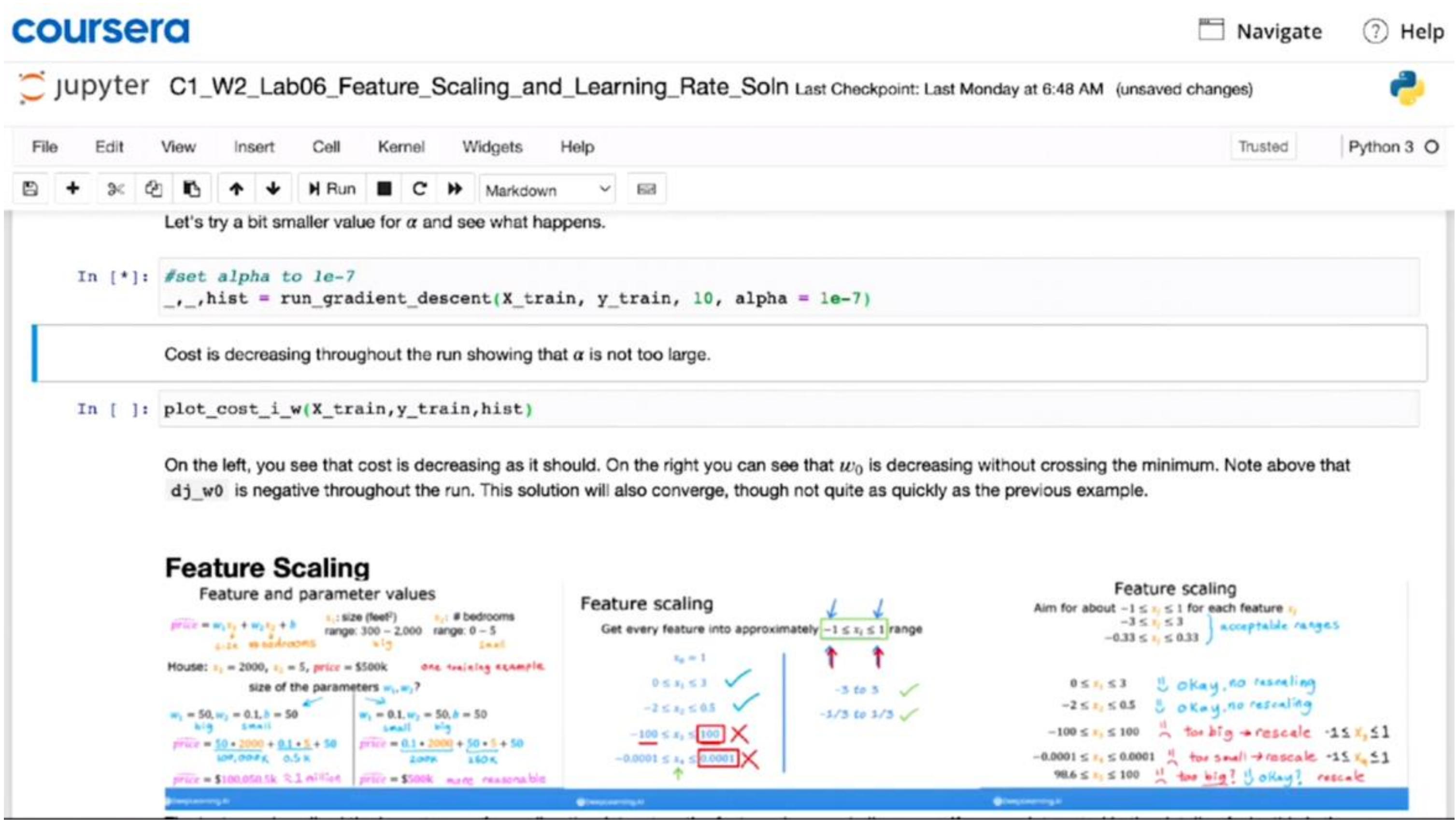

coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
Cost vs Iteration
Cost
iteration
Cost vs w[0]
w[0]
5:31 / 6:06

```
w,b found by gradient descent: w: [ 7.74e-02  8.27e-05 -1.06e-06 -4.20e-03], b: 0.00
```

Cost is decreasing throughout the run showing that alpha is not too large.

```
In [7]: plot_cost_i_w(X_train, y_train, hist)
```

On the left, you see that cost is decreasing as it should. On the right, you can see that w_0 is still oscillating around the minimum, but it is decreasing each iteration rather than increasing. Note above that `dj_dw[0]` changes sign with each iteration as `w[0]` jumps over the optimal value. This alpha value will converge. You can vary the number of iterations to see how it behaves.


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Let's try a bit smaller value for α and see what happens.

```
In [*]: #set alpha to 1e-7
        _,_,hist = run_gradient_descent(X_train, y_train, 10, alpha = 1e-7)
```

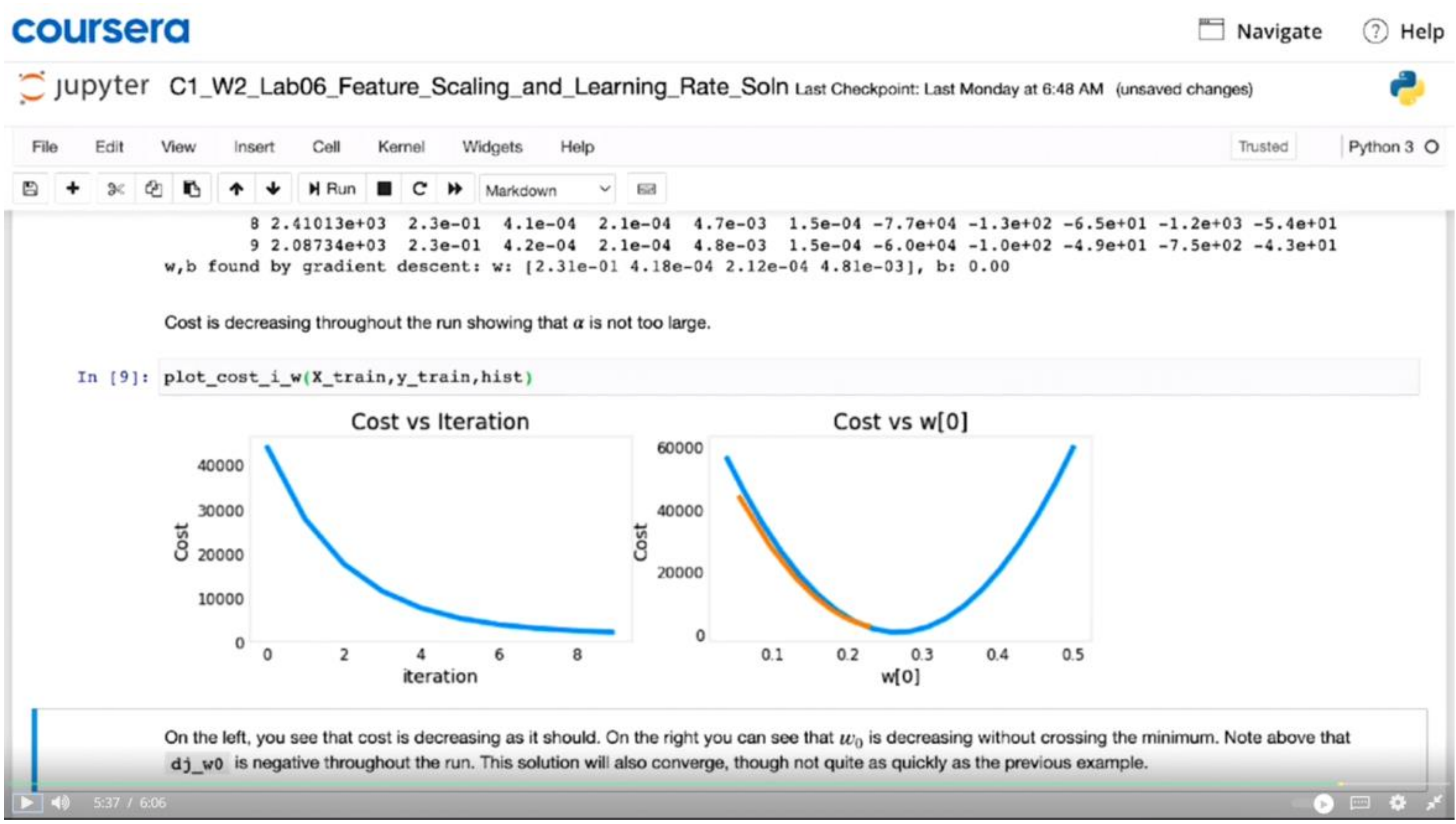
Cost is decreasing throughout the run showing that α is not too large.

```
In [ ]: plot_cost_i_w(X_train,y_train,hist)
```

On the left, you see that cost is decreasing as it should. On the right you can see that w_0 is decreasing without crossing the minimum. Note above that `dj_w0` is negative throughout the run. This solution will also converge, though not quite as quickly as the previous example.

Feature Scaling

Feature and parameter values
$\widehat{price} = w_1x_1 + w_2x_2 + b$
size
#bedrooms
x_1: size (feet²) range: 300 – 2,000 big
x_2: # bedrooms range: 0 – 5 small
House: $x_1 = 2000$, $x_2 = 5$, $price = \$500k$ one training example
size of the parameters w_1, w_2?
$w_1 = 50, w_2 = 0.1, b = 50$ big small
$\widehat{price} = 50 * 2000 + 0.1 * 5 + 50$
100,000K 0.5K
$\widehat{price} = \$100,050.5k$ ≈ 1 million
$w_1 = 0.1, w_2 = 50, b = 50$ small big
$\widehat{price} = 0.1 * 2000 + 50 * 5 + 50$
200K 250K
$\widehat{price} = \$500k$ more reasonable
Feature scaling
Get every feature into approximately $-1 \le x_j \le 1$ range
$x_0 = 1$
$0 \le x_1 \le 3$ ✓
$-2 \le x_2 \le 0.5$ ✓
$-100 \le x_3 \le 100$ ✗
$-0.0001 \le x_4 \le 0.0001$ ✗
-3 to 3 ✓
-1/3 to 1/3 ✓
Feature scaling
Aim for about $-1 \le x_j \le 1$ for each feature x_j
$-3 \le x_j \le 3$
$-0.33 \le x_j \le 0.33$
acceptable ranges
$0 \le x_1 \le 3$ okay, no rescaling
$-2 \le x_2 \le 0.5$ okay, no rescaling
$-100 \le x_3 \le 100$ too big → rescale $-1 \le x_3 \le 1$
$-0.0001 \le x_4 \le 0.0001$ too small → rescale $-1 \le x_4 \le 1$
$98.6 \le x_5 \le 100$ too big? okay? rescale


coursera
Navigate
Help
jupyter C1_W2_Lab06_Feature_Scaling_and_Learning_Rate_Soln Last Checkpoint: Last Monday at 6:48 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

```
        8 2.41013e+03  2.3e-01  4.1e-04  2.1e-04  4.7e-03  1.5e-04 -7.7e+04 -1.3e+02 -6.5e+01 -1.2e+03 -5.4e+01
        9 2.08734e+03  2.3e-01  4.2e-04  2.1e-04  4.8e-03  1.5e-04 -6.0e+04 -1.0e+02 -4.9e+01 -7.5e+02 -4.3e+01
w,b found by gradient descent: w: [2.31e-01 4.18e-04 2.12e-04 4.81e-03], b: 0.00
```

Cost is decreasing throughout the run showing that α is not too large.

```
In [9]: plot_cost_i_w(X_train,y_train,hist)
```

Cost vs Iteration
Cost
iteration
0
10000
20000
30000
40000
0 2 4 6 8
Cost vs w[0]
Cost
w[0]
0
20000
40000
60000
0.1 0.2 0.3 0.4 0.5

On the left, you see that cost is decreasing as it should. On the right you can see that w_0 is decreasing without crossing the minimum. Note above that `dj_w0` is negative throughout the run. This solution will also converge, though not quite as quickly as the previous example.

T, Z, f, Chi , Poisson, Binomial Distributions

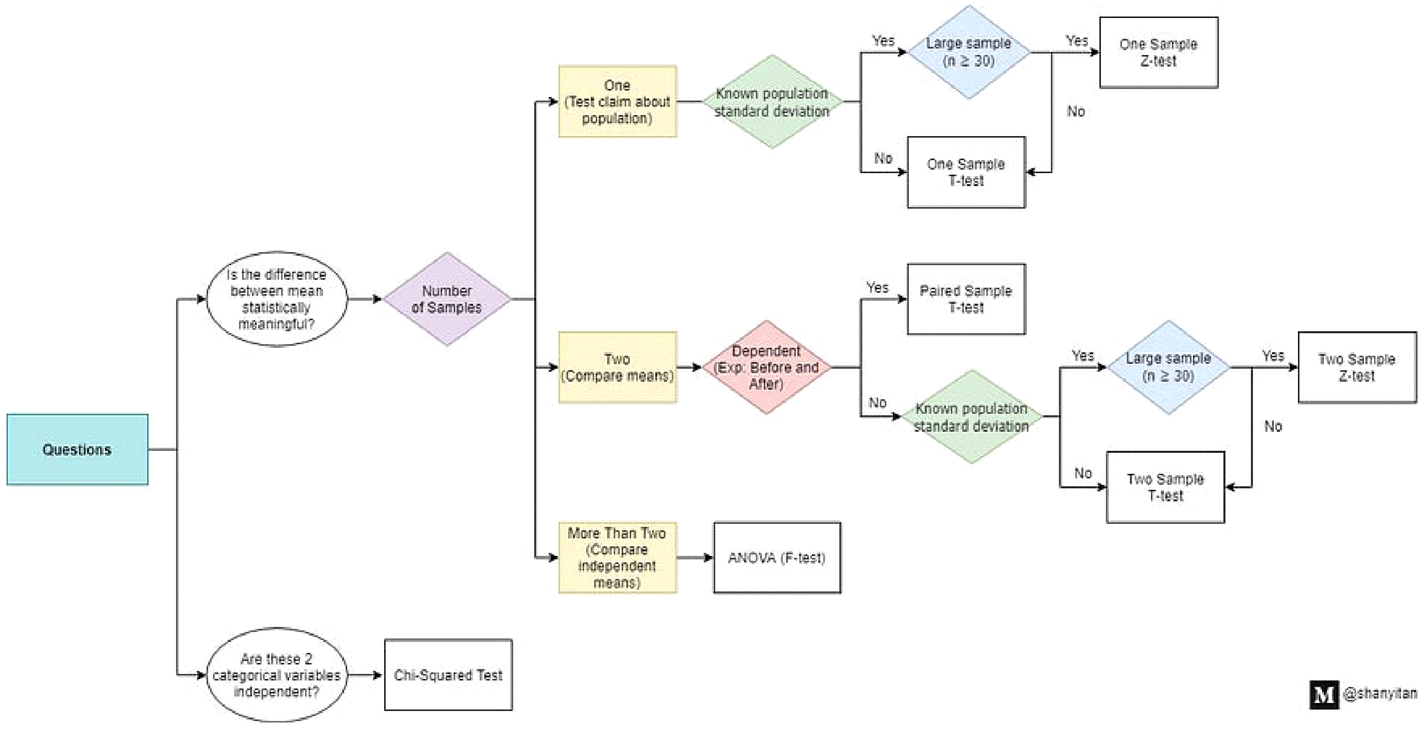
Wednesday, April 19, 2023 11:54 AM

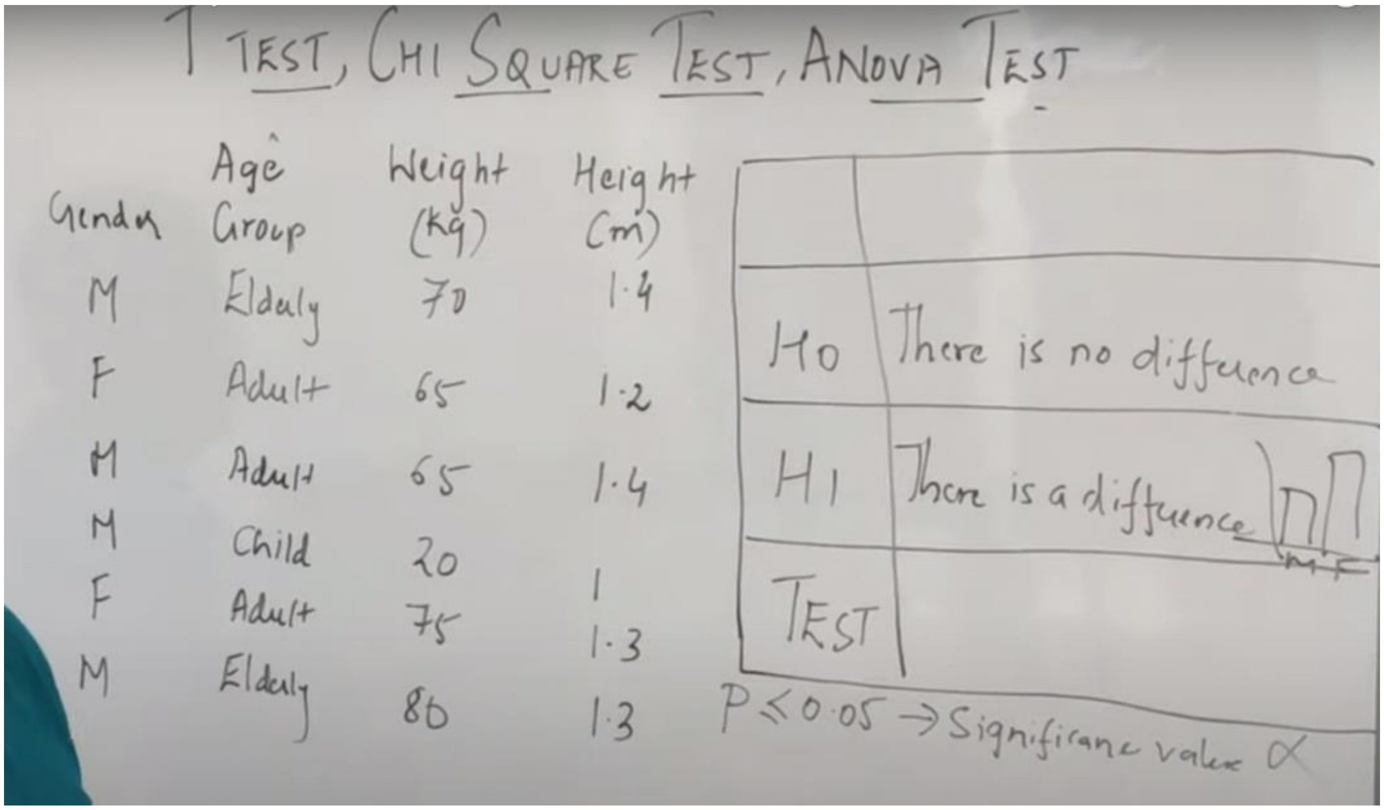
When to apply which statistical test

1 categorical feature	One sample proportion test
2 categorical feature	Chi Square
1 continuous feature	T - test (sample size <30)
1 continuous feature	Z - test (sample size >=30)
2 continuous features	Pearson correlation
More than 2 group	ANOVA

Types of Tests:

1. Z test {Comparison of Mean}
2. t Test {Comparison of Mean}
3. ANOVA test (F test) { Analysis of Variance}
4. Chi Square {Comparison between two categorical variables}


Questions
Is the difference between mean statistically meaningful?
Number of Samples
One (Test claim about population)
Known population standard deviation
Yes
Large sample (n ≥ 30)
Yes
One Sample Z-test
No
No
One Sample T-test
Two (Compare means)
Dependent (Exp: Before and After)
Yes
Paired Sample T-test
No
Known population standard deviation
Yes
Large sample (n ≥ 30)
Yes
Two Sample Z-test
No
No
Two Sample T-test
More Than Two (Compare independent means)
ANOVA (F-test)
Are these 2 categorical variables independent?
Chi-Squared Test
@shanyitan



T TEST, CHI SQUARE TEST, ANOVA TEST

Gender	Age Group	Weight (kg)	Height (cm)
M	Elderly	70	1.4
F	Adult	65	1.2
M	Adult	65	1.4
M	Child	20	1
F	Adult	75	1.3
M	Elderly	80	1.3

H_0	There is no difference
H_1	There is a difference
TEST	

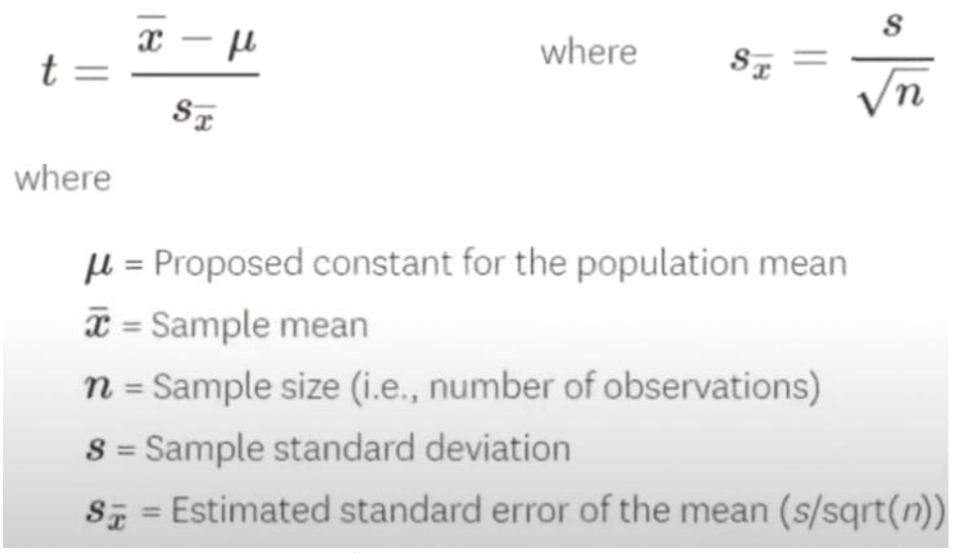
$P \leq 0.05 \rightarrow$ Significance value α

For the given data if we have to find the difference

1. When we have only 1 categorical feature Gender and null hypothesis is there is no difference between Male and Female gender proportion
   We can use one sample proportion test and calculate the p- value if P-value <0.05 then we reject the null hypothesis, It means that the probability of null hypothesis is true is less than 5% so we accept the alternate hypothesis
2. When we have 2 categorical features Gender and age and null hypothesis is there is no difference between Male and Female based on their age group
   We can use Chi square test to calculate p value
3. When we have 1 continuous variable height and the null hypothesis is the average height is 1.3
   We can use T test to calculate p -value
4. When we have 2 continuous variables and the null hypothesis is there is no relationship
   we can use Pearson Correlation
5. If we have numerical variable and categorical variable and the categorical variable has 2 category then T test and more than 2 category then ANOVA

T -test
T test is a type of inferential statistics which is used to determine if there is a significant difference between the mean of 2 groups
There is 2 types of t test

a. One sampled t-test: tell us whether means of the sample and the population are different



$$t = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad \text{where} \quad s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

where

μ = Proposed constant for the population mean
$\bar{x}$ = Sample mean
n = Sample size (i.e., number of observations)
s = Sample standard deviation
$s_{\bar{x}}$ = Estimated standard error of the mean (s/sqrt(n))

b. Two sampled t-test : It compares the mean of two independent groups, this is applied to the whole population

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

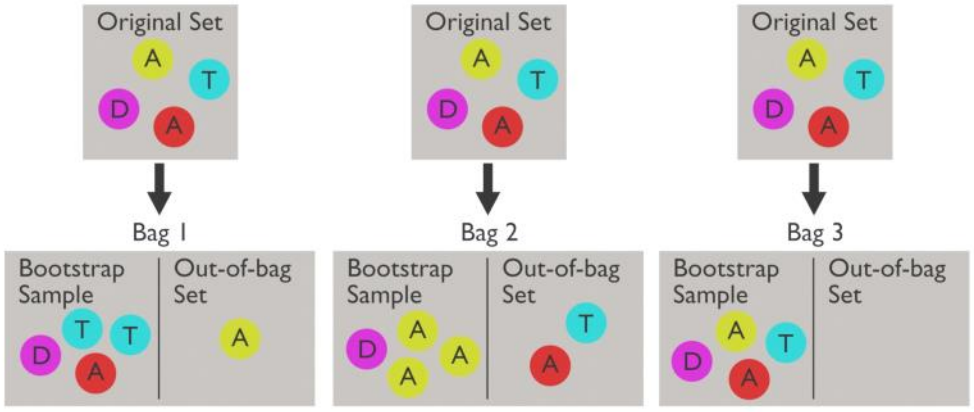
$$s^2 = \frac{\sum_{i-1}^{n_1}(x_i - \bar{x}_1)^2 + \sum_{j-1}^{n_2}(x_j - \bar{x}_2)^2}{n_1 + n_2 - 2}$$

c. Paired t -test: When you want to check how different samples are from the same population group

OOB(Out of Bag)

Monday, May 8, 2023 2:37 PM

In bagging, when different samples are collected, no sample contains all the data but a fraction of the original dataset. There might be some data that are never sampled at all. The remaining data which are not sampled are called out of bag instances.



Out of Bag score or Out of bag error is the technique, or we can say it is a validation technique mainly used in the bagging algorithms to measure the error or the performance of the models in every epoch for reducing the total error of the models in the end.

Out of Bag Score: What is it?

In each step of the **bootstrapping**, a small part of the data points from the samples fed to the bottom learner is taken, and each bottom model makes predictions after being trained on the sample data. The prediction error on that sample is known as the out-of-bag error. The OOB score is the number of **correctly predicted data on OOB samples** taken for validation. It means that the more the error bottom model does, the Less the OOB score for the bottom model. Now, this OOB score is used as the error of the particular bottom models and depending upon this, the model's performance is enhanced.

Out of Bag Score: Why Use it?

Now a question may arise, why the OOB Score is required? What is the need for that?

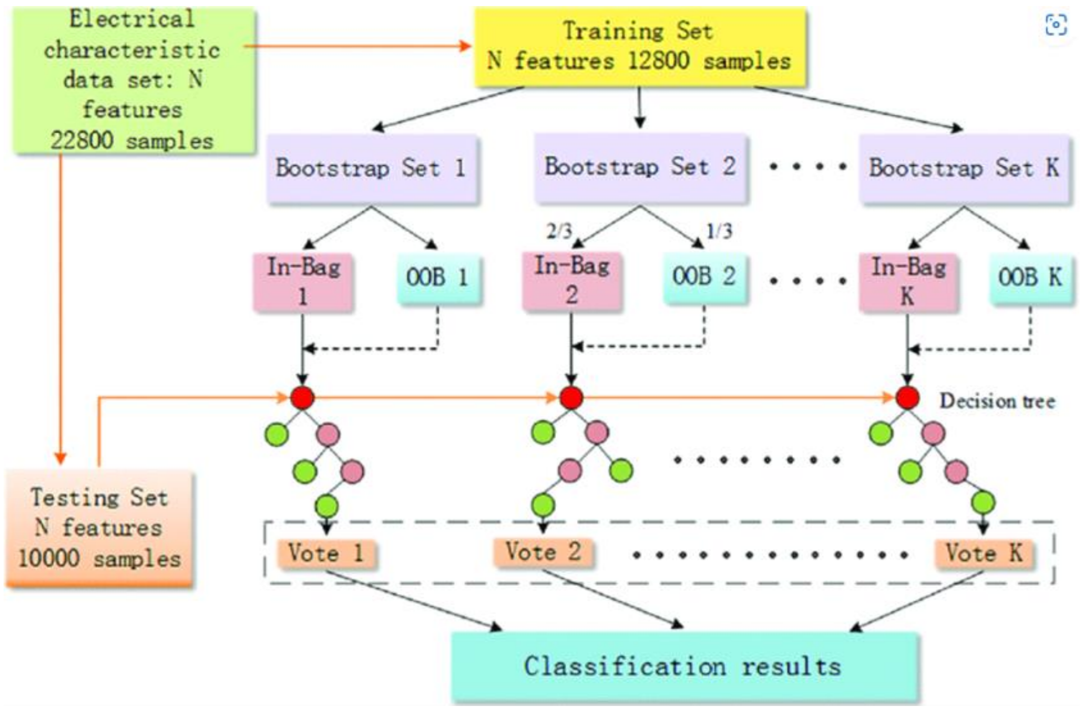
The OOB is calculated as the number of correctly predicted values by the bottom models on the validation dataset taken from the bootstrapped sample data. This OOB score helps the bagging algorithm understand the bottom **models' errors on anonymous data**, depending upon which bottom models can be hyper-tuned.

For example, a decision tree of full depth can lead to overfitting, so let's suppose we have a bottom model of the decision tree of the full depth and being overfitted on the dataset. Now in the case of overfitting, the error rate on the training data will be meager, but on the testing data, it will be very high. So the validation data will be taken from the bootstrapped sample, and the OOB score will be shallow. As the model is overfitting, the errors will be high on validation data which is entirely unknown and lead to the low OOB Score.

As we can see in the above example, the OOB score helps the model to understand the scenarios where the model is not behaving well and using which the final errors of the models can be reduced.

Out of Bag Score: How Does it Work?

Let's try to understand how the OOB score works, as we know that the OOB score is a measure of the correctly predicted values on the validation dataset. The validation data is the **sub-sample of the bootstrapped sample data** fed to the bottom models. So here, the validation data will be recorded for every bottom model, and every bottom model will be trained on the bootstrapped samples. Once all the bottom models are trained on the fed selection, the validation samples will be used to calculate the OOB error of the bottom models.



As we can see in the above image, the dataset sample contains a total of 1200 rows, out of which the three bootstrapped samples will be fed to the bottom model for training. Now from the bootstrap samples, 1,2, and 3, the small part or validation part of the data will be taken as OOB samples. These bottom models will be trained on the other part of the **bootstrap samples**, and once trained, the OOB samples will be used to predict the bottom models. Once the bottom models predict the OOB samples, it will calculate the OOB score. The exact process will now be followed for all the bottom models; hence, depending upon the OOB error, the model will **enhance its performance**.

To get the **OOB Score** from the **Random Forest Algorithm**, Use the code below.

```
from sklearn.trees import RandomForestClassifier
rfc = RandomForestClassifier(oob_score=True)
rfc.fit(X_train,y_train)
print(rfc.oob_score_)
```

The Advantages of the OOB Score

1. Better Performance of the model

As the OOB score indicates the error of the bottom models based on the validation data set, the model can get an idea about the mistakes and enhance the model's performance.

2. No Data Leakage

Since the validation data for OOB samples are taken from the bootstrapped samples, the data is being used only for prediction, which means that the data will not be used for the training, which ensures that the data will not leak. The model will not see the validation data, which is quite good as the OOB score would be genuine if the data is kept secret.

3. Better For small datasets

OOB score is an excellent approach if the dataset size is small to medium. It performs so well on a small dataset and returns a better predictive model.

The Disadvantage of the OOB Score

1. High Time Complexity

As validation samples are taken and used for validating the model, it takes a lot of time to do the same process for multiple epochs; hence, the time complexity of the OOB score is very high.

2. Space Complexity

As some of the validation data is collected from bootstrap samples, now there will be more splits of the data in the model, which will result in more need of space to save and use the model.

2. Poor performance on Large Dataset

OOB score needs to perform better on large datasets due to space and time complexities.

Key Takeaways

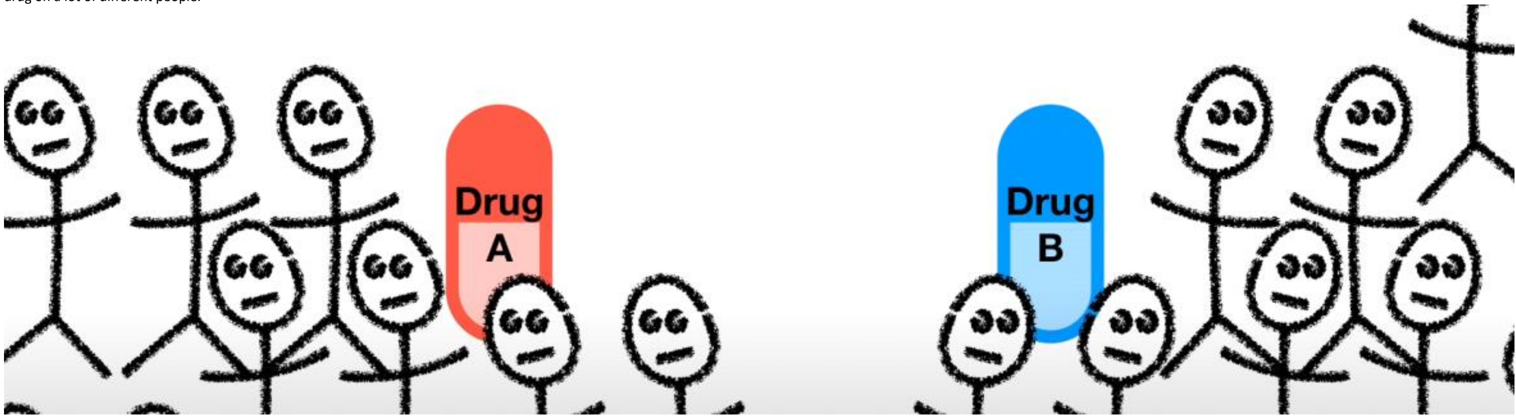
1. OOB error is the measurement of the error of the bottom models on the validation data taken from the **bootstrapped** sample
2. OOB score helps the model understand the bottom model's error and returns **better predictive models**.
3. OOB score performs so well on **small datasets** but [or large ones.
4. OOB score has **high time complexity** but ensures **no data leakage**.

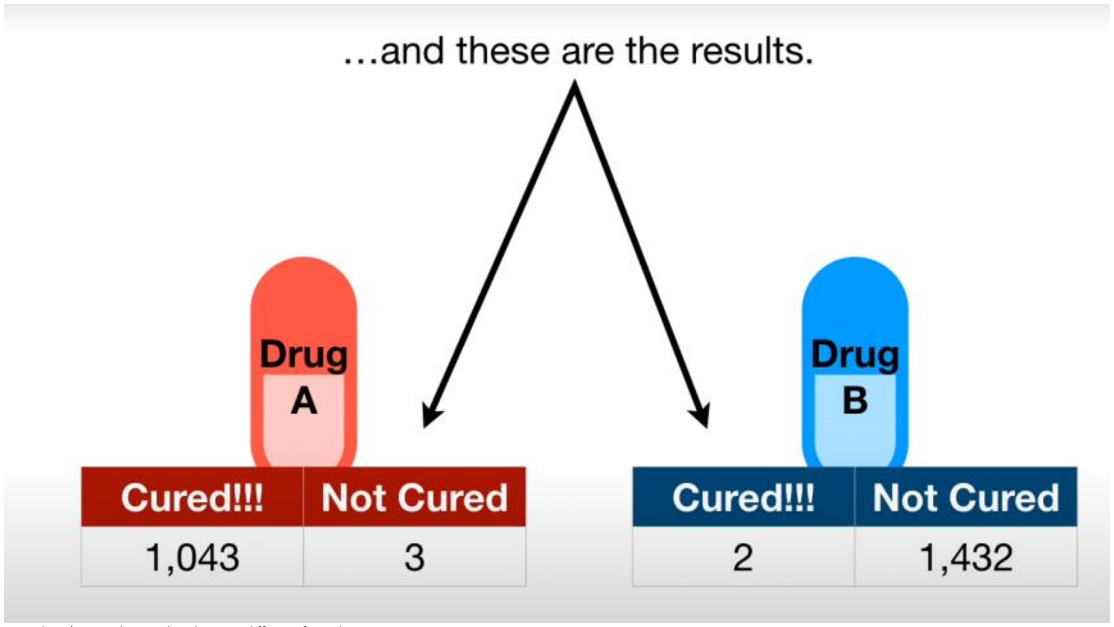
Hypothesis testing, P- value

Monday, July 10, 2023 12:18 PM

Hypothesis testing:
Rather than get stressed out over a large number of possible hypothesis that we could test to see if two drugs are different we simply use the null hypothesis to determine if there is a difference.
If we take a lot more people and a lot more people taking drugs C which has shorter recovery time than people taking drug D, if we repeat the experiment so many times then it would be hard to imagine that the results were due to random things like everyone taking drugs had better diet or got more exercise than the people taking drug D. if the difference is not 0 then we can reject the null hypothesis
Or
In hypothesis testing we try to evaluate 2 mutually exclusive statement on population using sample data
null hypothesis: it treats everything same or equal, alternate hypothesis: it is opposite to the null hypothesis

P-value :
It is the probability for the null hypothesis to be true
Or
Imagine I have two drugs drug A and drug B And I want to know if drug A is better from drug B. So I give one person drug A and another person drug B. The one person drug A is cured. The one person drug B is not cured. Can we conclude drug A is better than drug B Drug B? Drug B may have failed because of lot of different reason. There are a lot of weird, random things that can help when doing a test. Means that we need to test the drug on more than just one people each. So now we test the drug on a lot of different people.

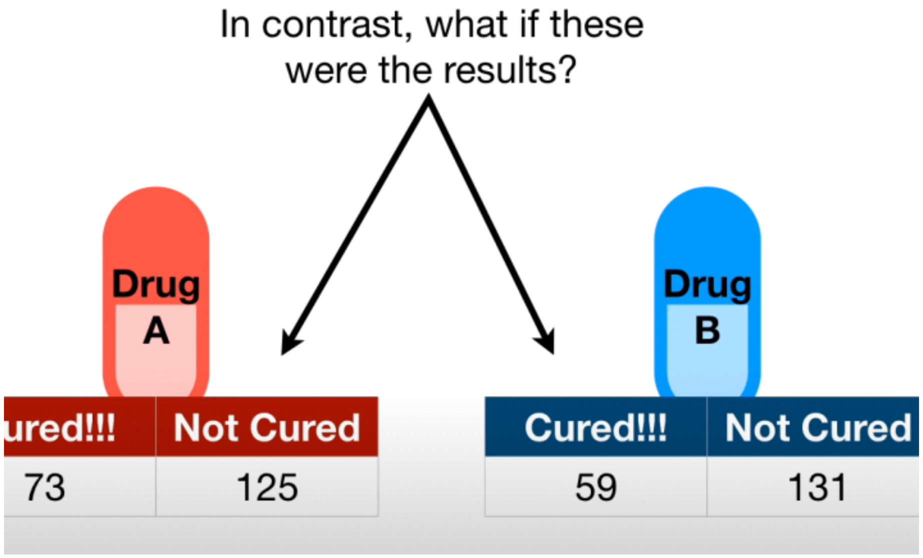



...and these are the results.
Drug A
Drug B

Cured!!!	Not Cured
1,043	3

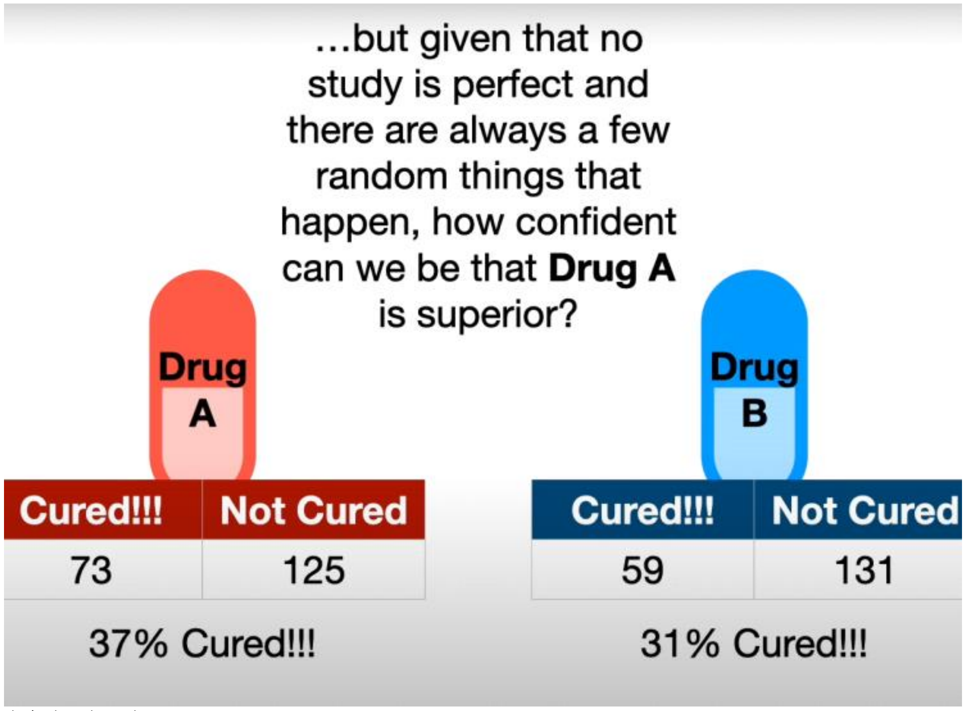
Cured!!!	Not Cured
2	1,432

So in this it's very obvious that drug A is different from drug B .


In contrast, what if these were the results?

ured!!!	Not Cured
73	125

Cured!!!	Not Cured
59	131


...but given that no study is perfect and there are always a few random things that happen, how confident can we be that **Drug A** is superior?

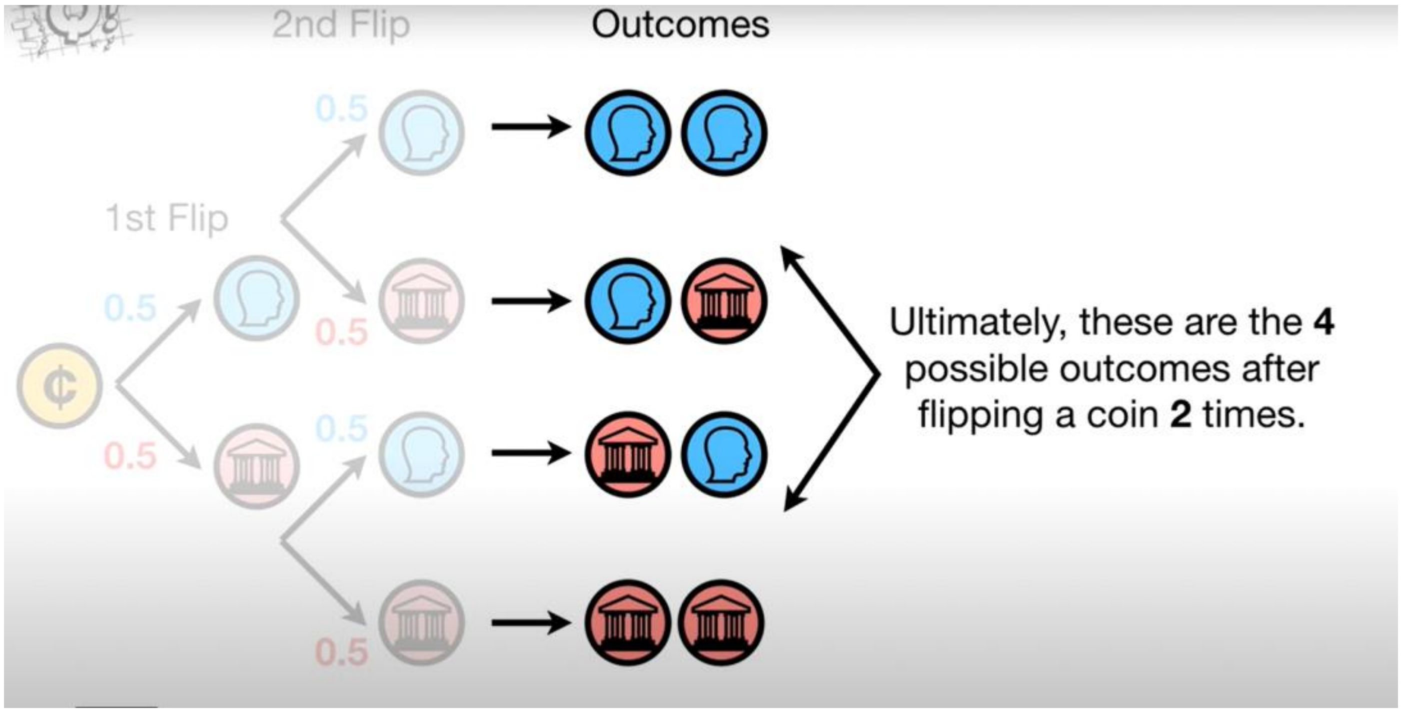
Cured!!!	Not Cured
73	125

37% Cured!!!

Cured!!!	Not Cured
59	131

31% Cured!!!

That's where the P values comes in.
P values are number between zero and one Which tells how confident we should be that drug A is different from drug B. The closer the P value is to 0 the more confident we have that the drug A and drug B is different.
So the question is how small a P value have to be before we sufficiently confident that drug A is different from a drug B?
In another words what threshold can we use to make a good decision ?
in practice a commonly used threshold is 0.05. This means that we are 95% confident that drug and drug B is different only 5% of the time this might be wrong . When the values are less than 5% those are called false positive case.
When P-value < 0.05 then we decide that drug A is different from drug B
if P value is 0.24 then we are not sure that drug A is different from drug B
In statistics the idea of determining if these drugs are the same or not is called hypothesis testing the null hypothesis is that the drugs are the same and the P value decides if we should reject the null hypothesis or not. the small P value help us to decide whether the both drugs are same or not but it doesn't tell us how different they are , the difference can be tiny or huge
there are two types of P values 1 sided and two sided. Two sided P values are most common
suppose we have a coin and we flipped it once and we got the head. then i flipped it again and got the head second time So here the hypothesis is even I got the two heads in a row but my coin is no different from normal coin
these are the outcomes if we flip the coin two times



Ultimately, these are the **4** possible outcomes after flipping a coin **2** times.