**Berkeley 14 - 16 AI/ML Certificate Program Modules 14 - 16**

Prepare Journey Data, generalize and mask entries.!

Slack: Here's the announcement pointing to the Slack space.

Topic: Student Success Coaching and Slack Workspace Now Available!

(emeritus.org)

<https://student.emeritus.org/courses/4765/discussion_topics/257315>

* 941884802
* 639519956
* 622799832
* 576043581
* 614049009
* 594801975
* 478128440
* 894489611
* 750116839
* 537618010

**Savio’s Colab:**

<https://colab.research.google.com/drive/1m640WoWTbTgw0ymZoIw9K0wyCfMXtbYH#scrollTo=YdnBR4JQ_MMA>

One 1:1 Meeting per participant across the period of Weeks 12-15

June 8 - July 5

One 1:1 Meeting per participant across the period of Modules 21-23

August 17 - September 6

<https://student.emeritus.org/courses/4765/pages/how-to-schedule-a-1-1-session?module_item_id=999645>

<https://student.emeritus.org/courses/4765/pages/capstone-project-overview?module_item_id=999644>

<https://student.emeritus.org>

**An outline of the BH-PCMLAI program calendar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Module #** | **Module Title** | **Week #** | **Date** |
| **0** | Program Orientation | 0 | **Wednesday, March 02, 2022** |
| **1** | Introduction to Machine Learning | 1 | **Wednesday, March 09, 2022** |
| **2** | Fundamentals of Machine Learning | 2 | **Wednesday, March 16, 2022** |
| **3** | Introduction to Data Analysis | 3 | **Wednesday, March 23, 2022** |
| **4** | Fundamentals of Data Analysis | 4 | **Wednesday, March 30, 2022** |
| **5** | Practical Application 1 | 5 | **Wednesday, April 06, 2022** |
| Break Week |  |  | **Wednesday, April 13, 2022** |
| **6** | Clustering and Principal Component Analysis (PCA) | 6 | **Wednesday, April 20, 2022** |
| **7** | Linear and Multiple Regressions | 7 | **Wednesday, April 27, 2022** |
| **8** | Feature Engineering and Overfitting | 8 | **Wednesday, May 04, 2022** |
| **9** | Model Selection and Regularization | 9 | **Wednesday, May 11, 2022** |
| **10** | Time Series Analysis and Forecasting | 10 | **Wednesday, May 18, 2022** |
| **11** | Practical Application 2 | 11 | **Wednesday, May 25, 2022** |
| Break Week |  |  | **Wednesday, June 01, 2022** |
| **12** | Classification and k-Nearest Neighbors (KNN) | 12 | **Wednesday, June 08, 2022** |
| **13** | Logistic Regression | 13 | **Wednesday, June 15, 2022** |
| **14** | Decision Trees | 14 | **Wednesday, June 22, 2022** |
| **15** | Gradient Descent and Optimization | 15 | **Wednesday, June 29, 2022** |
| **16** | Support Vector Machines (SVMs) | 16 | **Wednesday, July 06, 2022** |
| **17** | Practical Application 3 | 17 | **Wednesday, July 13, 2022** |
| Break Week |  |  | **Wednesday, July 20, 2022** |
| **18** | Natural Language Procession (NLP) | 18 | **Wednesday, July 27, 2022** |
| **19** | Recommendation Systems | 19 | **Wednesday, August 03, 2022** |
| **20** | Capstone 1 | 20 | **Wednesday, August 10, 2022** |
| **21** | Ensemble Techniques (GBM, XGB, and Random Forest) | 21 | **Wednesday, August 17, 2022** |
| **22** | Deep Neural Networks 1 | 22 | **Wednesday, August 24, 2022** |
| **23** | Deep Neural Networks 2 | 23 | **Wednesday, August 31, 2022** |
| **24** | Capstone 2 | 24 | **Wednesday, September 07, 2022** |

 ————— o —————

**————— o —————**

**Module 14**

**Decision Trees**

1. Build a decision tree manually
2. Train a decision tree model with desired hyperparameters using scikit-learn
3. Visualize a decision tree
4. Evaluate overfitting of decision trees
5. Evaluate decision tree splits
6. Implement the decision tree algorithm
7. Compare the performance of different grid search algorithms

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3353899?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3353899/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3353900?wrap=1)

Diagram

Description automatically generated

**Notes:**

Glossary

**Branch**

A connection between nodes

**Entropy**

The degree of impurity or uncertainty in a set of observations

**Internal Node**

A node that has two or more branches and can be split further

**Leaf Node**

A node that can no longer be split

**Pruning**

A set of techniques to ‘trim’ branches from the decision tree, making the tree faster to compute

**Root Node**

The beginning node of a decision tree

**Weighted Entropy**

The entropy of a node multiplied by the fraction of samples in that node

X2 = X[['flipper\_length\_mm', 'bill\_length\_mm']]

dtree = DecisionTreeClassifier(max\_depth = 2).fit(X2, y)

tree2 = export\_text(dtree, feature\_names = list(X2.columns))

### ANSWER CHECK

print(tree2)

sns.scatterplot(data=X2, x='flipper\_length\_mm', y='bill\_length\_mm', hue=y)

plt.axvline(x=206.50, color="black")

plt.axhline(y=43.35, xmin=0, xmax=0.575, color="black")

plt.axhline(y=40.85, xmin=0.575, xmax=1, color="black")

dtree3 = DecisionTreeClassifier(max\_depth = 3).fit(X2, y)

tree3 = export\_text(dtree3, feature\_names = list(X2.columns))

prediction = dtree3.predict(pd.DataFrame([[209, 41.2]], columns=X2.columns))

import ssl

ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context

**Module Issues:**

Codio 14.4 Problem 4: must to declare those variables:

**ent\_age1 = -(1 \* np.log2(1))**

**ent\_age2 = -(1/4 \* np.log2(1/4) + 3/4 \* np.log2(3/4))**

Codio 14.4 Problem 5: parameter should be **criterion = 'entropy'**

Codio 14.6 Problem 3: Do not include the last element in ccp\_alphas in the for loop: **for i in ccp\_alphas[:-1]**

Codio 14.6 Problem 4: Change to **plt.step(ccp\_alphas, depths, '--o')**

Codio 14.6 marked as Codio 14.7 in the activity.

**Quizes:**

A decision tree is a tree of questions that should be answered arbitrarily to yield a predictive classification. : False

*You are correct! The answer “*False*” is correct because a decision tree is a tree of questions that should be answered in sequence to yield a predictive classification.*

The decision boundaries of a logistic regression model are linear. : True

*You are correct! The answer “*True*” is correct because the decision boundaries of a logistic regression model are linear.*

Chart, scatter chart

Description automatically generated

Consider this plot with petal\_length on the x-axis and petal\_width on the y-axis. For a decision rule “petal\_width < 0.75 and petal\_length < 2” equal to yes, what class would be chosen? : setosa

*You are correct! The answer “*setosa*” is correct because in this range of decision rules, setosa is the only class present, thus making setosa the best prediction.*

What is the correct Python statement to import a decision tree model from scikit-learn? : from sklearn import tree

*You are correct! The answer “*from sklearn import tree*” is correct because this is the correct Python statement to import decision tree models from scikit-learn.*

“d\_t=tree.DecisionTree(criterion=’entropy’)”

The given Python code is the correct code for building a decision tree classifier. : False

*You are correct! The answer “*False*” is correct because the correct function for building a decision tree classifier is* *“*DecisionTreeClassifier(criterion=’entropy’)*”.*

Which of the following is not a constructor used in “tree.plot\_tree()” function? : criterion

*You are correct! The answer “*criterion*” is correct because this is the constructor of the “*DecisionTreeClassifier()*” not for “*plot\_tree()” *function.*

Seaborn is the best library to visualize a decision tree. : False

*You are correct! The answer “*False*” is correct because the Python library used to visualize the decision tree plot with better layout is Graphviz.*

A node from a decision tree

|  |
| --- |
| petal\_width ≤ 1.75 |
| entropy = 1.0 |
| samples = 100 |
| value = [0, 50, 50] |
| class = versicolor |

Given a single node of a decision tree built with class\_names=[“a”,“b”,“c”], how many samples belong to class “a” at this node? : 0

*You are correct! The answer “*0*” is correct because the parameter ‘value’ at the specific node tells the number of samples belonging to each class. Since there are three classes (“*a*”, “*b*”, and “*c*”) and the ‘value’ parameter here says [0, 50, 50], that means there are 0 samples in class “*a*” (and 50 each in “*b*” and “*c*”).*

What is the correct Python function that can be used to measure the accuracy of a decision tree classifier? : accuracy\_score(predictions,original\_data)

*You are correct! The answer “*accuracy\_score(predictions,original\_data)*” is correct because this is the correct Python function used to measure the accuracy of a decision tree classifier.*

Scikit-learn decision trees always have perfect accuracy on the training data, except when there are samples from the same classes with the exact same features. : False

*You are correct! The answer “*False*” is correct because scikit-learn decision trees always have perfect accuracy on the training data except when there are samples from different classes with the exact same features.*

The training accuracy for a decision tree classifier can be 100% when the data points overlap exactly. : False

*You are correct! The answer “*False*” is correct because the training accuracy for a decision tree classifier can be 100% when the data points do not overlap exactly.*

A picture containing chart

Description automatically generated

Timeline

Description automatically generated

Consider the two decision trees above. Which of the plots seems to show overfitting? : Plot 1

*You are correct! The answer “*Plot 1*” is correct because it is highly complex, and the more complex the plot, the higher the chance of overfitting. Plot 1 seems to be more complex with more chances of overfitting.*

For decision tree classifiers, more features do not necessarily lead to overfitting. : True

*You are correct! The answer “*True*” is correct because more features do not lead to overfitting but can help in resolving the difference between the classes.*

The very first node of a decision tree is known as the leaf node. : False

*You are correct! The answer “*False*” is correct because the very first node of a decision tree is known as the root node.*

In the decision tree algorithm, after selecting the best feature “x” and the best split value “β”, the data is split into two nodes. One of the nodes has a rule where x < β. What is the rule for the other node? : x >= β

*You are correct! The answer “*x >= β*” is correct because after selecting “x” and “*β”,*the data is split into two nodes, one where x < β and one where*x >= β.

A node in a decision tree that has overlapping data points from different classes and thus cannot be split is called a “pure” node. : False

*You are correct! The answer “*False*” is correct because a node in the decision tree that has overlapping data points from different classes and thus cannot be split is called an ‘unsplittable’ node*.

A node from a decision tree

|  |
| --- |
| sepal\_length ≤ 5.45 |
| gini = 0.665 |
| samples = 110 |
| value = [24, 36, 40] |
| class = virginica |

For this given node of a decision tree classifier, what would the value of “p2” be? : 0.36

*You are correct! The answer “*0.36*” is correct because p2 would be calculated as 40/110 which comes out to be “0.36”.*

What is the entropy (S) for a node with p values of [“p0=0.31”, “p1=0.33”, “p2=0.36”]? : 1.58

*You are correct! The answer “*1.58*” is correct because with the given values, the formula for entropy gives*

*−0.31 log2 0.31 − 0.33 log2 0.33 − 0.36 log2 0.36, which comes out to be 1.58.*

A node in a decision tree where data is evenly split between two classes has entropy 1. : True

*You are correct! The answer “*True*” is correct because with evenly split data for two classes, the formula for entropy is*

*−0.5 log2 0.5 − 0.5 log2 0.5, which is equal to 1.*

The weighted entropy (WS) of a node is its entropy scaled by the fraction of samples in that node. : True

*You are correct! The answer “*True*” is correct because the weighted entropy (WS) of a node is its entropy multiplied by the fraction of samples in that node divided by the total samples.*

The weighted entropy of the root node is equal to its entropy. : True

*You are correct! The answer “*True*” is correct because the weighted entropy is given as the entropy multiplied by the fraction of samples in that node, and the root node's fraction is 1 (n/n). Therefore, the entropy and the weighted entropy of the root node will always be the same.*

Diagram

Description automatically generated with medium confidence

From the given information, what would ΔWS be for the root node? : 0.46

*You are correct! The answer “*0.46*” is correct because the ΔWS is given as the WS of the node minus the WS of the child nodes, which is 0.67 − (0.16 + 0.046) = 0.46.*

The decision tree classifier chooses the best feature and the best split such that the ΔWS is minimized. : False

*You are correct! The answer “*False*” is correct because the decision tree classifier chooses the best feature and the best split such that the ΔWS is maximized.*

The decision tree classifier can use L1 and L2 regularization. : False

*You are correct! The answer “*False*” is correct because decision trees do not have parameters that act as weights, so the algorithm cannot use L1 or L2 regularization.*

The default option for min\_samples\_split is two. : True

*You are correct! The answer “*True*” is correct because the default option for min\_samples\_split is 2.*

**Quiz Problems:**

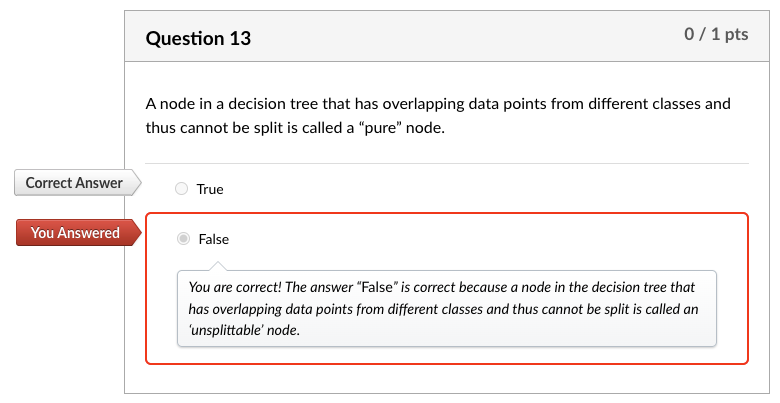
Wrong answers set in the template for 4 questions, please see the attachments and correct the answer template as well as my grade. Thanks.

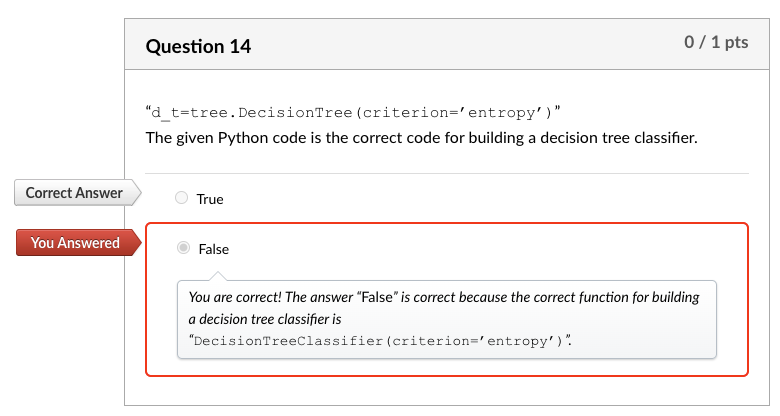
Graphical user interface, text, application, Teams

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated





**Discussion 14.1: You Already Use Decision Trees**

Decision Tree for enjoying a (state) park visit

Do you want to visit a close by park?

Yes No

| |

Do you want to see peacocks? Do you want to drive less than 3 hours?

Yes No Yes No

| | | |

Go to Mayfield Park and Nature Preserve Emma Long Metropolitan Park Blue hole Park

How far you want to drive?

Do you want to drive less than 2 hours?

**Yes**

Do you want amenities and restaurants

Yes : Muller Park

Do you want to exercise but still in downtown?

Yes: Lady Bird Lake @ Cesar Chavez

No: McKinney Falls

**No:**

Do you want to stay in close proximity of a downtown?

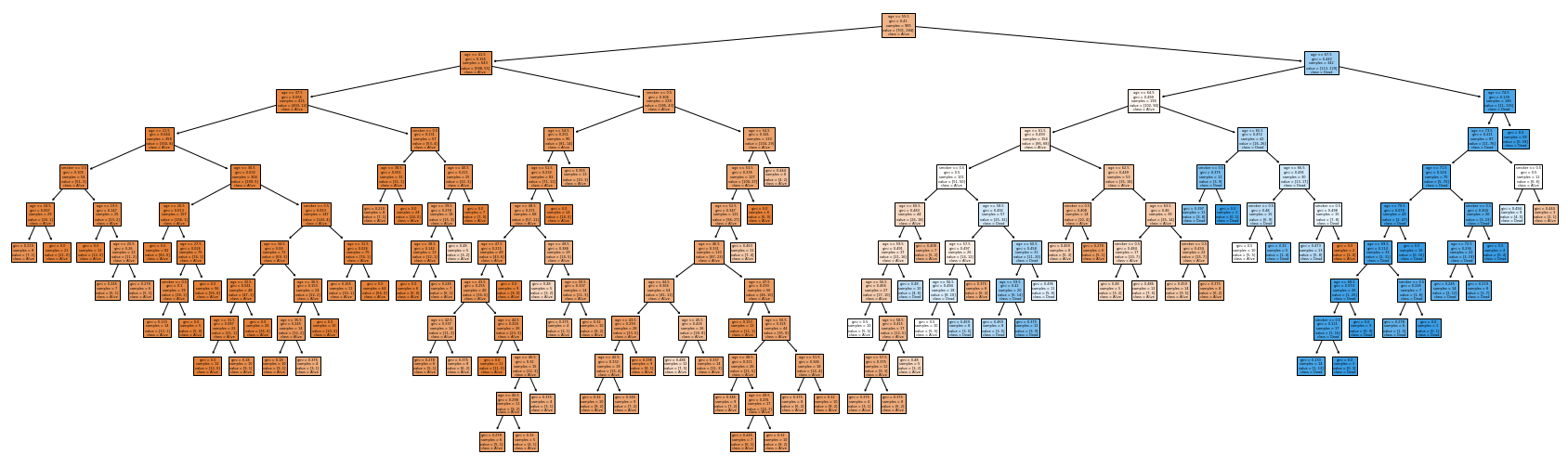
Yes: Blue Hole Regional Park

No:

My decision tree is for choosing a park for recreational activities, sometimes I go restaurants after those places, other times just hiking, sometimes we do some country side driving. This is a simple decision tree what I want to do that day, yellow is close by parks, green is moderate drive, red is farther park:

Diagram

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

Text, letter

Description automatically generated

Chart, line chart

Description automatically generated

**Try-It Activity 14.1: Generating Overfit Models - Section B**

**Dataset Analysis**

Before iterating through depths until the maximum, I analyzed the dataset. Here is my findings about it:

There are no null values, 1314 entries, 2 string columns.

**Feature Overlapping**

I got curious about feature overlapping for different classes, sure enough there are 72 feature overlapping cases affecting ~11% of the entire population, or model cannot perform for those:

# number of overlapping feature with different classes

data\_size = data.groupby(['smoker', 'age', 'outcome']).size().groupby(['smoker', 'age']).size().reset\_index()

data\_size[data\_size[0]>1]

**Data Preparation**

The rest is transforming smoker column to integer 1/0 binary values, checking if the dataset is balanced, it is 72% and 28% distribution per outcome target column.

y.value\_counts(normalize=True)

**Maximum Depth for Upper Bound**

I ran a default model to get maximum depth for upper bound:

# get maximum depth with default parameters:

max\_tree = DecisionTreeClassifier(random\_state=93).fit(X\_train, y\_train)

max\_depth = max\_tree.get\_depth()

**Iteration for Depth**

Iterating the model over while incrementing the max\_depth parameter to analyze overfitting

# Try out all depths to capture scores

train\_accs = []

test\_accs = []

for d in range(1,max\_depth+1,1):

#create decision tree and fit it

tree = DecisionTreeClassifier(random\_state=93, max\_depth=d).fit(X\_train, y\_train)

# populate train and test accuracy lists

train\_accs.append(tree.score(X\_train, y\_train))

test\_accs.append(tree.score(X\_test, y\_test))

**Accuracy Scores**

# plot scores

plt.subplots(figsize = (10, 7))

plt.step(range(1,max\_depth+1,1), train\_accs, '-o', label = 'Train')

plt.step(range(1,max\_depth+1,1), test\_accs, '-o', label = 'Test')

plt.plot(np.argmax(test\_accs)+1, max(test\_accs), 'ro', markersize = 12, alpha = 0.4, label = 'Best Score')

plt.xticks(range(1,max\_depth+1,1), rotation = 0)

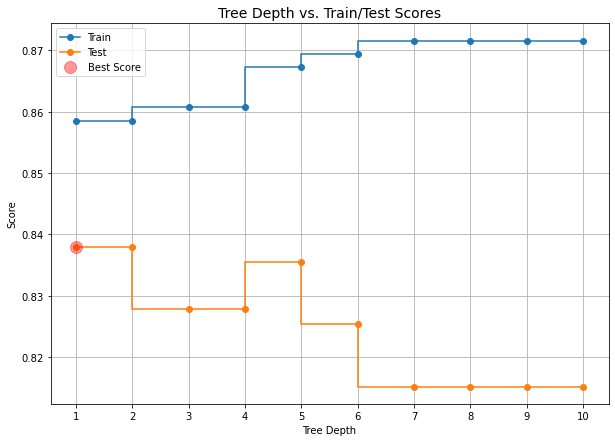
plt.legend()

plt.grid()

plt.xlabel('Tree Depth')

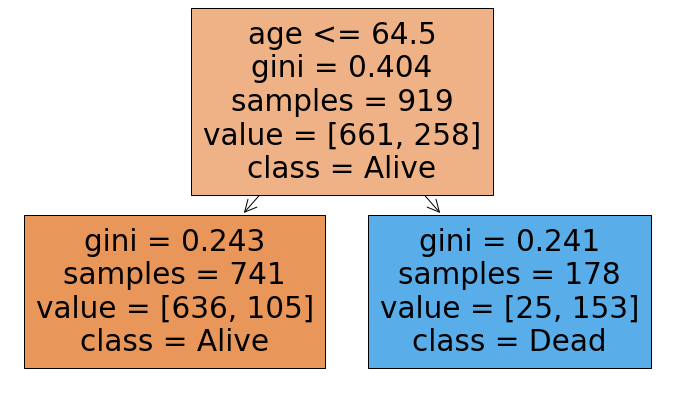
plt.ylabel('Score')

plt.title('Tree Depth vs. Train/Test Scores', fontsize=14)

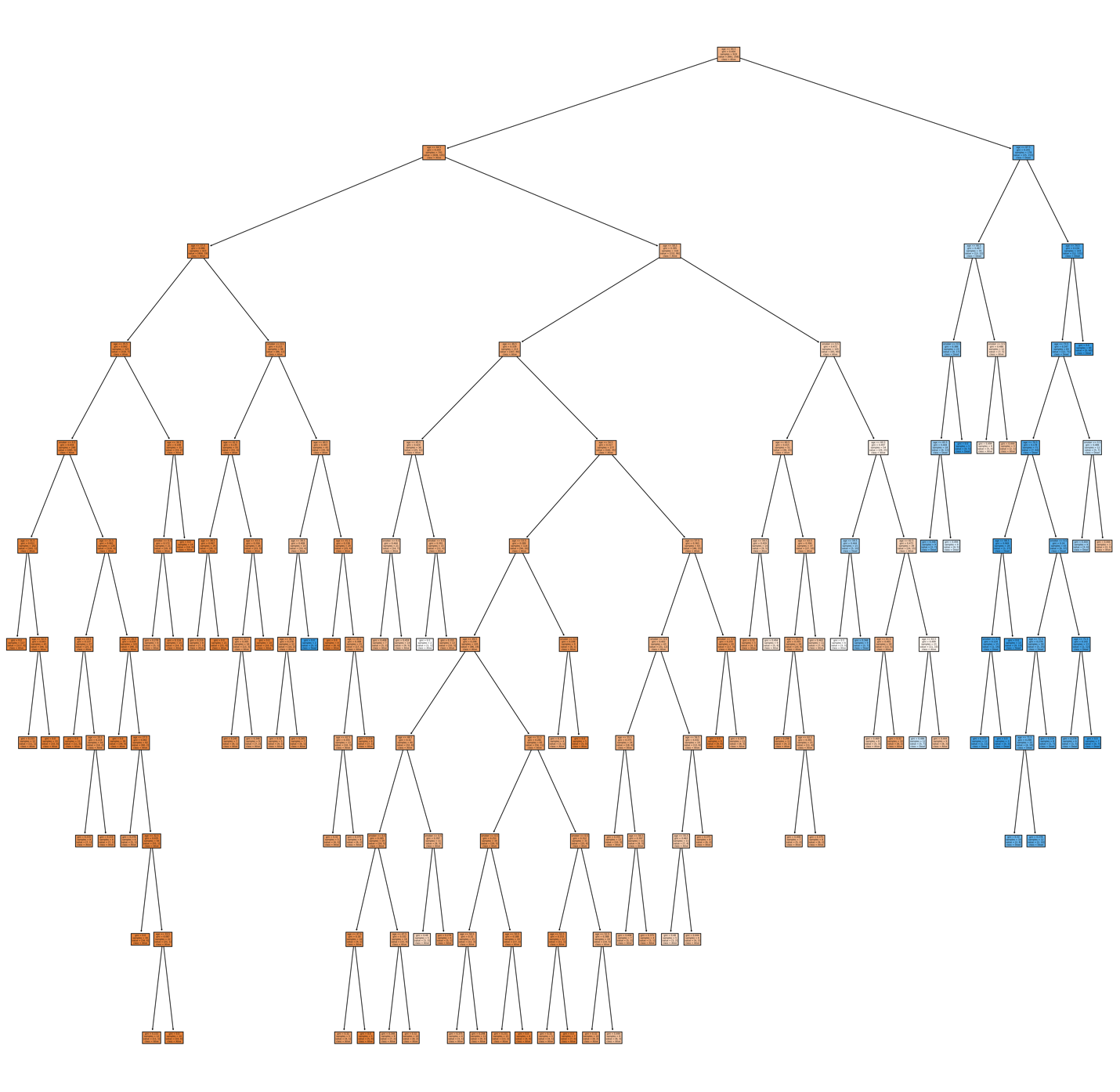


Overfitting begins after max\_depth=1 in this case as test scores decreasing. Please note, depth=1 and 2 scores are the same.

**Optimum Tree:**



**Maximum Depth Tree:**



**Conclusion**

Best accuracy score is an indicator we can spot where overfitting begins on the plot per test accuracy scores. The score degraded when increasing the tree depth, the best score is about 84%, I computed the feature overlapping as 11% which may yield false classification, the best we can get from the model is 89% anyway. In contrast to that, the train accuracy increases with tree-depth increase which is another sign of overfitting.

**Try-It Activity 14.2: Grid Searching Decision Trees - Section B**

I used the same dataset we used in the previous try-t activity. Same analysis, data is imbalanced 72% and 28% distribution per *outcome* target column, 72 feature overlapping, etc.

**Parameters for Grid Search**

# Put some values to try out

params = {'max\_depth': [2, 3, 4, 5, 6, 7, 10],

'min\_samples\_split': [0.1, 0.2, 0.05],

'criterion': ['gini', 'entropy'],

'min\_samples\_leaf': [2, 3, 4, 5]

}

**Model**

# initialize

dtree = DecisionTreeClassifier(random\_state=93)

**Comparing Grid Searches**

I built 4 grid search for:

* GridSearchCV
* RandomizedSearchCV
* HalvingGridSearchCV
* HalvingRandomSearchCV

# GridSearchCV

grid = GridSearchCV(dtree, param\_grid=params).fit(X\_train, y\_train)

grid\_train\_acc = grid.score(X\_train, y\_train)

grid\_test\_acc = grid.score(X\_test, y\_test)

best\_params = grid.best\_params\_

### Results

print(f'Training Accuracy: {grid\_train\_acc: .2f}')

print(f'Test Accuracy : {grid\_test\_acc: .8f}')

print(f'Best parameters of tree: {best\_params}')

Training Accuracy: 0.86

Test Accuracy : 0.83797468

Best parameters of tree: {'criterion': 'gini', 'max\_depth': 2, 'min\_samples\_leaf': 2, 'min\_samples\_split': 0.1}

# RandomizedSearchCV

rgrid = RandomizedSearchCV(dtree, param\_distributions=params).fit(X\_train, y\_train)

rgrid\_train\_acc = rgrid.score(X\_train, y\_train)

rgrid\_test\_acc = rgrid.score(X\_test, y\_test)

rbest\_params = rgrid.best\_params\_

### Results

print(f'Training Accuracy: {rgrid\_train\_acc: .2f}')

print(f'Test Accuracy : {rgrid\_test\_acc: .8f}')

print(f'Best parameters of tree: {rbest\_params}')

Training Accuracy: 0.86

Test Accuracy : 0.83797468

Best parameters of tree: {'min\_samples\_split': 0.2, 'min\_samples\_leaf': 2, 'max\_depth': 7, 'criterion': 'gini'}

# HalvingGridSearchCV

hgrid = HalvingGridSearchCV(dtree, param\_grid=params).fit(X\_train, y\_train)

hgrid\_train\_acc = hgrid.score(X\_train, y\_train)

hgrid\_test\_acc = hgrid.score(X\_test, y\_test)

hbest\_params = hgrid.best\_params\_

### Results

print(f'Training Accuracy: {hgrid\_train\_acc: .2f}')

print(f'Test Accuracy : {hgrid\_test\_acc: .8f}')

print(f'Best parameters of tree: {hbest\_params}')

Training Accuracy: 0.86

Test Accuracy : 0.83797468

Best parameters of tree: {'criterion': 'entropy', 'max\_depth': 3, 'min\_samples\_leaf': 4, 'min\_samples\_split': 0.05}

# HalvingRandomSearchCV

hrgrid = HalvingRandomSearchCV(dtree, param\_distributions=params).fit(X\_train, y\_train)

hrgrid\_train\_acc = hrgrid.score(X\_train, y\_train)

hrgrid\_test\_acc = hrgrid.score(X\_test, y\_test)

hrbest\_params = hrgrid.best\_params\_

### Results

print(f'Training Accuracy: {hrgrid\_train\_acc: .2f}')

print(f'Test Accuracy : {hrgrid\_test\_acc: .8f}')

print(f'Best parameters of tree: {hrbest\_params}')

Training Accuracy: 0.86

Test Accuracy : 0.83797468

Best parameters of tree: {'min\_samples\_split': 0.1, 'min\_samples\_leaf': 3, 'max\_depth': 2, 'criterion': 'gini'}

Their train and test accuracy scores came out the same for all execution results but each search method picked different best parameters.

I used *mean\_fit\_time* in cv\_results\_ to computed cumulated elapsed time for the plots when I need time comparison among all:

# Time spent

# set metrics!

grid\_options=['GridSearchCV','RandomizedSearchCV','HalvingGridSearchCV','HalvingRandomSearchCV']

train\_accs = [grid\_train\_acc, rgrid\_train\_acc, hgrid\_train\_acc, hrgrid\_train\_acc]

test\_accs = [grid\_test\_acc, rgrid\_test\_acc, hgrid\_test\_acc, hrgrid\_test\_acc]

elapsed\_times = [np.sum(grid.cv\_results\_['mean\_fit\_time']), np.sum(rgrid.cv\_results\_['mean\_fit\_time']),

np.sum(hgrid.cv\_results\_['mean\_fit\_time']), np.sum(hrgrid.cv\_results\_['mean\_fit\_time'])]

depths = [best\_params['max\_depth'], rbest\_params['max\_depth'], hbest\_params['max\_depth'], hrbest\_params['max\_depth']]

splits = [best\_params['min\_samples\_split'], rbest\_params['min\_samples\_split'],

hbest\_params['min\_samples\_split'], hrbest\_params['min\_samples\_split']]

leaves = [best\_params['min\_samples\_leaf'], rbest\_params['min\_samples\_leaf'],

hbest\_params['min\_samples\_leaf'], hrbest\_params['min\_samples\_leaf']]

# plot accuracy and time elapsed

fig, ax = plt.subplots(1, 2, figsize = (15, 8))

ax[0].plot(grid\_options, train\_accs, '--o', label = 'Training Accuracy')

ax[0].plot(grid\_options, test\_accs, '--o', label = 'Testing Accuracy')

ax[0].plot(grid\_options, [max(test\_accs)]\*len(grid\_options), 'ro', markersize = 12, alpha = 0.4, label = 'Best Score')

ax[0].tick\_params(axis='x', rotation=90)

ax[0].set\_xlabel('Grid Search Options')

ax[0].set\_ylabel('Accuracy Score')

ax[0].set\_title('Grid Search Options vs. Accuracy Score all is best')

ax[0].legend()

# time plot

ax[1].plot(grid\_options, elapsed\_times, '--o', label = 'Elapsed Time')

ax[1].tick\_params(axis='x', rotation=90)

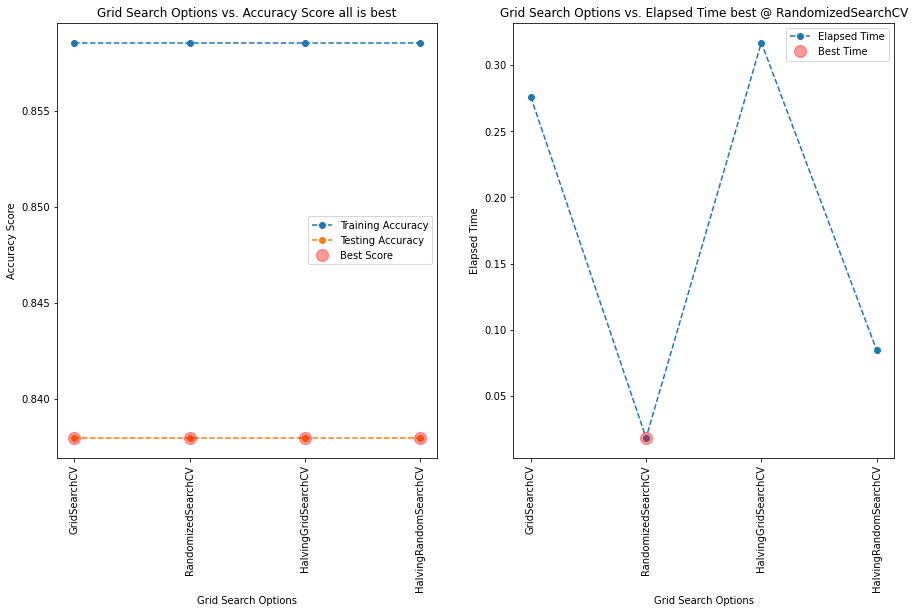
ax[1].set\_xlabel('Grid Search Options')

ax[1].set\_ylabel('Cumulative Elapsed Time')

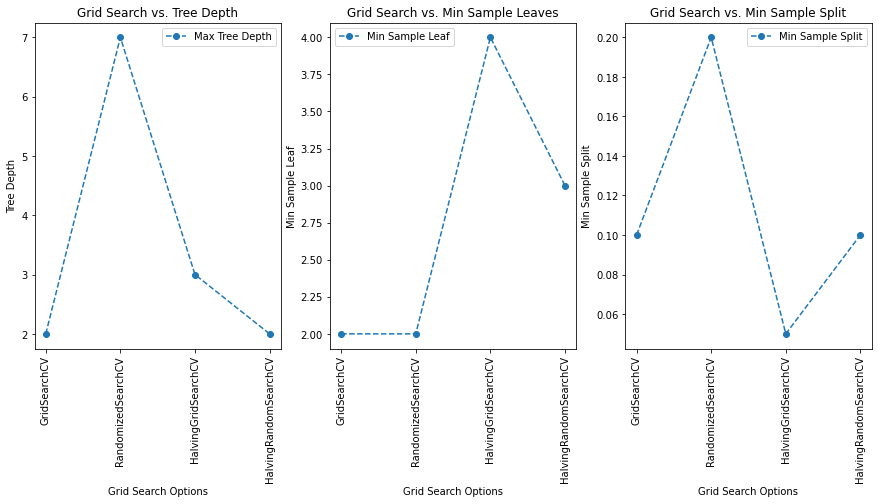
ax[1].set\_title(f'Grid Search Options vs. Elapsed Time best @ {grid\_options[np.argmin(elapsed\_times)]}')

ax[1].plot(np.argmin(elapsed\_times), min(elapsed\_times), 'ro', markersize = 12, alpha = 0.4, label = 'Best Time')

ax[1].legend()



Consensus on all methods for best score, however, RandomizedSearchCV is winner in elapsed time! However, GridSearchCV and HalvingGridSearchCV are significantly higher cumulative times than RandomizedSearchCV and HalvingRandomSearchCV.



Best max\_depth=2, two methods GridSearchCV and HalvingRandomSearchCV achieved it. Best min\_samples\_leaf=2, by GridSearchCV and RandomizedSearchCV. Best min\_samples\_split=0.05 by HalvingGridSearchCV.

**Conclusion**

Best parameters from GridSearchCV and HalvingRandomSearchCV have the minimum max\_depth=2 which reduces complexity.

Timeline

Description automatically generated

Also, the other best parameters are the same {'min\_samples\_split': 0.1, 'max\_depth': 2, 'criterion': 'gini'} except 'min\_samples\_leaf': 3 by HalvingRandomSearchCV and 'min\_samples\_leaf': 2 by GridSearchCV.

Both produced the exact same decision tree but HalvingRandomSearchCV is more than 3 times faster than GridSearchCV.

 ————— o —————

**Module 15**

**Gradient Descent and Optimization**

1. Compute x and/or y for each iteration of gradient descent
2. Explain how learning rate and starting guess affect the convergence of gradient descent
3. Optimize a single-parameter linear regression model from scratch
4. Recognize convex one-dimensional and two-dimensional functions
5. Compute the gradient of a two-dimensional function
6. Use gradient descent to optimize a nonlinear two-dimensional regression model
7. Use stochastic gradient descent to optimize a nonlinear two-dimensional regression model
8. Compare the convergence behavior of gradient descent with stochastic gradient descent
9. Identify the degrees of bias and variance in a model
10. Identify the relationship between bias, variance, and model complexity

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3390438?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3390438/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3390449?wrap=1)

**Program learning outcomes addressed this module:**

* Apply real-world tools to model and analyze real-world data
* Communicate foundational concepts about AI/ML
* Draw useful conclusions from real-world data
* Identify the best ML model to solve a problem (Models: Classification, regression, time series analysis)

**Optimizing Linear Regression**

As previously discussed, linear regression assumes only one independent variable and a linear relationship between the independent (X) and dependent (Y) variables.Linear regression determines the relationship between the two variables by fitting a linear equation to the observed data. The best fit can be defined by the hypothesis equation, where w0 and w1 are weighted so that they are optimized to fit the best line and reduce cost (loss). Two primary methods are used to minimize the cost: Gradient descent and normal equation. Gradient descent can find the value of w0 and w1 using an iterative process to minimize the overall cost. In contrast, normal equation can find the weights (w) for which the cost is minimal.

**The Bias–Variance Tradeoff**

Prediction errors can be broken down into two main subcomponents: bias and variance. Bias and variance are key parameters that need to be tuned while training a machine learning model.

**Bias Error**

A bias error is the difference between a model’s predictions and the actual value. In this type of error, the model ignores training data and oversimplifies the model without learning the patterns.

**Variance Error**

A model’s ability to predict a given data point or value tells you how widely your data is spread. During this type of error, the model pays a lot of attention to training data, to the point that it memorizes the data rather than learning from it. As a result, models with a high variance error have difficulty generalizing on unseen data.

The bias–variance tradeoff describes the tension between bias-introduced and variance-introduced errors.

**Glossary**

**Bias Error**

The difference between a model’s predictions and the actual values

**Convexity**

A measure of the curvature, or the degree of the curve, where a line drawn between two points will have all points between the two endpoints under the line

**Gradient Descent**

An algorithm used to find a local minimum/maximum of a given function

**Stochastic Gradient Descent**

A technique that is an approximation of gradient descent

**Variance Error**

A model’s ability to predict a given data point or value that tells you how widely the data is spread

Gradient Descent is used in Liner Regression models, not for any other methodology!

**Notes:**

**Batch gradient descent**, also known as vanilla gradient descent, calculates the errors for each example in the training dataset, but the model is only updated once all training examples have been evaluated. The whole process is referred to as a training epoch.

**Stochastic gradient descent (SGD**) updates the parameters for every training example in the dataset, meaning each example’s parameters are updated one at a time. In some cases, this can make SGD faster than batch gradient descent.

**Mini-batch gradient descent** is the preferred method because it combines the concepts of batch gradient descent and SGD. It splits the training dataset into small batches and updates each batch individually. A balance is thus created between the robustness of SGD and the efficiency of batch gradient descent.

convex aka concave up!

M = (y2 - y1) / (x2 - x1)

t1s = ex2.iloc[:, 1]

t0s = ex2.iloc[:, 0]

theta1 = np.linspace(0, 20, 100)

theta0 = np.linspace(0, 20, 100)

T0, T1 = np.meshgrid(theta0, theta1)

fig = plt.figure(figsize=(16, 6))

ax = fig.add\_subplot(1, 2, 1, projection = '3d')

ax.plot3D(t0s, t1s, mse(t0s, t1s), '-->', color = 'red')

ax.plot\_wireframe(T0, T1, mse(T0, T1), alpha = 0.2)

ax2 = fig.add\_subplot(1, 2, 2)

ax2.contour(T0, T1, mse(T0, T1), levels = 30)

ax2.plot(t0s, t1s, mse(t0s, t1s), color = 'red')

ax2.set\_xlim(0, 20)

ax2.set\_ylim(0, 20)

**Module Issues:**

**Codio Activity 15.7 Problem 3**: Hidden Test uses *mse\_grad* student solution!

***for*** *i* ***in*** *range(1000):*

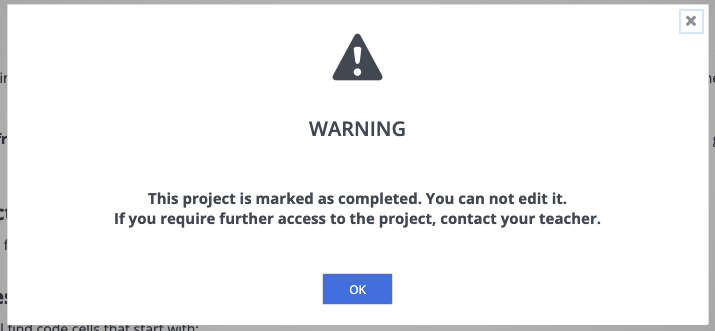
*thetas\_.append(theta\_)*

*theta\_ = theta\_ - lr\*mse\_grad(theta\_, X\_, y)*

**Codio Activity 15.8** Plot broken projection = “3”d

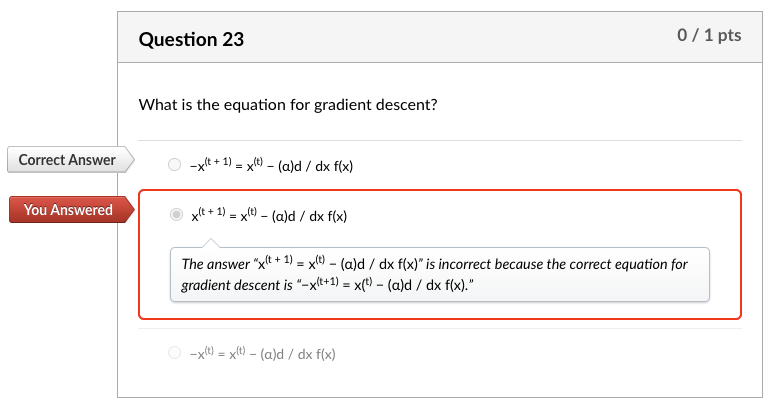
**Codio Activity 15.9 Problem 2**: typo in print(***sgd\_defaults***)

**Codio Activity 15.10** Reset option is not given

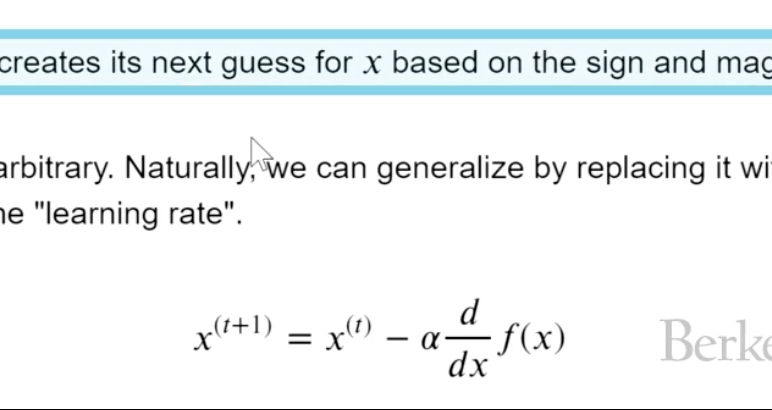


**End of Module Quiz Problems:**

The answer sheet template is wrong for a question, please see the attachment and correct the answer template as well as my grade. Thanks.



What is the equation for gradient descent?



**Quizes:**

Gradient descent is a technique used to minimize functions. : True

*You are correct! The answer “*True*” is correct because the gradient descent (GD) is an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function.*

Consider the following Python function:

minimize(arbitrary,x0)

What is the second constructor used as? : The starting point for the function to minimize

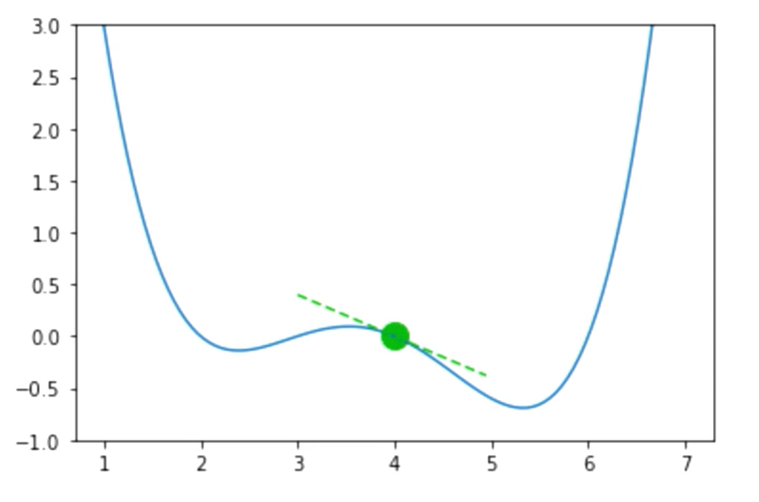
*You are correct! The answer “*The starting point for the function to minimize*” is correct because it represents the initial guess as the starting value to minimize the function.*

If the derivative of a function is negative, that means the function is (blank). : decreasing

*You are correct! The answer “*decreasing*” is correct because a function is decreasing if a derivative of that function is negative.*

In gradient descent, the point where there is a minima or maxima of the function is where the derivative is (blank). : zero

*You are correct! The answer “z*ero*” is correct because when the derivative of a function is zero that point is the minima or maxima of the function.*



Given the above graph, suppose the guess for a function input is x = 4, f(x) = 0.0, and f’(x) = −0.4. Should you increase or decrease x for your next guess? : Increase

*You are correct! The answer “*Increase*” is correct because if the derivative of a function is negative, that means the function is decreasing, so you increase the x.*

What is the equation for gradient descent? : x(t + 1) = x(t) − (α)d / dx f(x)

*You are correct! The answer “*x(t + 1) = x(t) − (α)d / dx f(x)*” is correct because this is the equation for gradient descent.*

The value alpha (α) in the gradient descent function is known as learning rate. : True

*You are correct! The answer “*True*” is correct because it is known as the learning rate.*

The learning rate alpha (α) captures how quickly gradient descent learns the minimum. A small alpha moves slowly but has a high chance of overshooting. : False

*You are correct! The answer “*False*” is correct because a small alpha means small jumps toward the minimum, which have a lower chance of overshooting.*

Gradient descent always gives the global minimum. : False

*You are correct! The answer “*False*” is correct because if the function has a local minimum and the algorithm’s initial guess (which is the starting point) is close to the minimum, it can get stuck at the local minimum.*

The gradient descent is guaranteed to find the global minimum if the function has convexity. : True

*You are correct! The answer “*True*” is correct because the gradient descent is guaranteed to find the global minimum if the function has convexity.*

Under what circumstances is the loss function f said to be convex? : If a line is drawn between two points on a curve, all values on the curve must be on or below the line

*You are correct! The answer “*If a line is drawn between two points on a curve, all values on the curve must be on or below the line*” is correct because this is the condition for a function to be convex.*

Consider the following the graph:

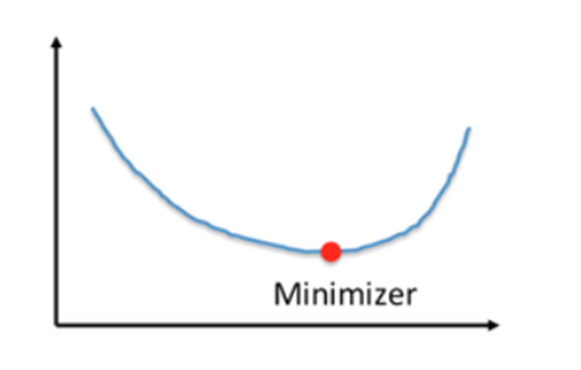
Chart, line chart

Description automatically generated

Is it convex or nonconvex? : Nonconvex

*You are correct! The answer “*Nonconvex*” is correct because the condition “if a line is drawn between two points on a curve, all values on the curve must be on or below the line” must be satisfied for a function to be convex. This plot does not satisfy this condition.*

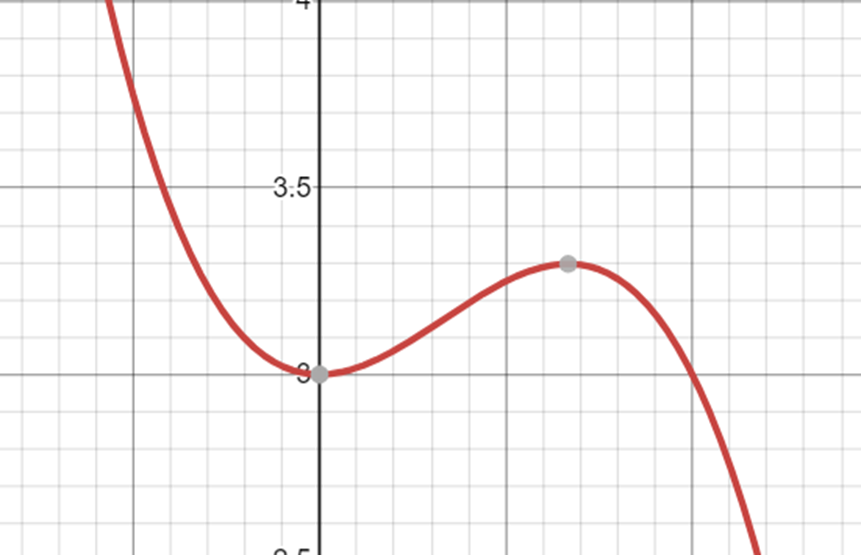
Consider the following graph:



Is it convex or nonconvex? : Convex

*You are correct! The answer “C*onvex*” is correct because the condition “if a line is drawn between two points on a curve, all values on the curve must be on or below the line” must be satisfied for a function to be convex. This plot satisfies this condition.*

Consider the following graph:



Is it convex or nonconvex?

*You are correct! The answer “*Nonconvex*” is correct because the condition “if a line is drawn between two points on a curve, all values on the curve must be on or below the line” must be satisfied for a function to be convex. This plot does not satisfy this condition.*

The function for mean squared error can be represented as mean((y\_hat - y\_obs)\*\*2). : True

*You are correct! The answer “*True*” is correct because this is the function for mean squared error.*

For a model Tip = θ1 + θ2 \* bill, the MSE loss function cannot be used to predict the optimal θ1 and θ2 values. : False

*You are correct! The answer “*False*” is correct because the MSE loss function can be used to predict the optimal θ1 and θ2 values.*

**def**mse\_loss(theta, X, y\_obs):     y\_hat **=** theta[0] **\*** X.iloc[:, 0] **+** theta[1] **\*** X.iloc[:, 1]

*#can also write in matrix form:* *#y\_hat = X @ theta*

**return**np.mean((y\_hat **-** y\_obs) **\*\*** 2)

From the function in the Python code block, determine the correct call statement for the function. : mse\_loss(np.array([1,2]),X,y\_obs)

*You are correct! The answer “*mse\_loss(np.array([1,2]),X,y\_obs)*” is correct because this is the correct function call statement.*

**def**mse\_loss(theta, X, y\_obs):     y\_hat **=** theta[0] **\*** X.iloc[:, 0] **+** theta[1] **\*** X.iloc[:, 1]

*#can also write in matrix form:* *#y\_hat = X @ theta*

**return**np.mean((y\_hat **-** y\_obs) **\*\*** 2)

From the function in the Python code block, determine the correct representation for a single argument function. : def mse\_loss\_single\_arg(theta):

    X = tips\_with\_bias[["bias", "total\_bill"]]

    y\_obs = tips["tip"]

    return mse\_loss(theta, X, y\_obs)

*You are correct! The answer “*def mse\_loss\_single\_arg(theta):

    X = tips\_with\_bias[["bias", "total\_bill"]]

    y\_obs = tips["tip"]

    return mse\_loss(theta, X, y\_obs)*” is correct because this is the correct representation for a single argument function.*

Given a function f(θ0,θ1) with two variables θ0 and θ1, what would the formula for the gradient of this 2D function be? : df / d(θ0)i + df / d(θ1)j

*You are correct! The answer “*df / d(θ0)i + df / d(θ1)j*” is correct because the gradient of the 2D function is a partial derivative with respect to variable one and a partial derivative with respect to variable two.*

To reduce the loss function, how should the θ values be adjusted? *Check all that apply.* : *“*Increase all values of θ that have a negative partial derivative*” and “*Decrease all values of θ that have a positive partial derivative.*”*

*You are correct! The answers “*Increase all values of θ that have a negative partial derivative*” and “*Decrease all values of θ that have a positive partial derivative*” are correct because these are the techniques to adjust the value of*θ *to reduce loss function.*

Consider the following two-dimensional linear regression model:

fθ→(x→)=(x→)Tθ→=θ0x0+θ1x1

What would the squared loss be for the single prediction of a linear regression model provided below? ℓ(θ→,x→,yi) : (yi − θ0x0 − θ1x1)2

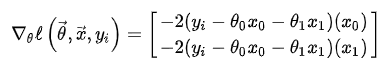
*You are correct! The answer “(yi − θ0x0 − θ1x1)2” is correct because this is the correct representation of squared loss for a single prediction of a linear regression model.*(yi − θ0x0 − θ1x1)2

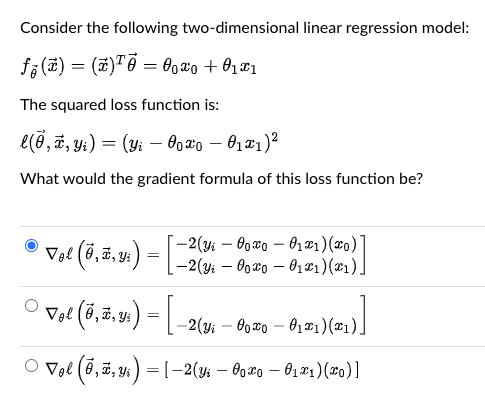
Consider the following two-dimensional linear regression model:

fθ→(x→)=(x→)Tθ→=θ0x0+θ1x1

The squared loss function is:ℓ(θ→,x→,yi)=(yi−θ0x0−θ1x1)2

What would the gradient formula of this loss function be? :





*You are correct! The answer “*

∇θℓ(θ→,x→,yi)=[

|  |
| --- |
| **−2(y**i**−θ**0**x**0**−θ**1**x**1**)(x**0**)** |
| **−2(y**i**−θ**0**x**0**−θ**1**x**1**)(x**1**)** |

]

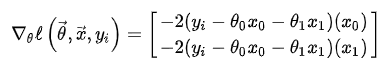
*” is correct because this is the correct formula for the gradient of this loss function.*

Consider the given gradient formula:

∇θℓ(θ→,x→,yi)=[

|  |
| --- |
| **−2(y**i**−θ**0**x**0**−θ**1**x**1**)(x**0**)** |
| **−2(y**i**−θ**0**x**0**−θ**1**x**1**)(x**1**)** |

]



Is this the Python function to compute these gradient values?

**def** mse\_gradient(theta, X, y\_obs):     """Returns the gradient of the MSE on our data for the given theta"""     x0 = X.iloc[:, 0]     x1 = X.iloc[:, 1]     dth0 = np.mean(-2 **\*** (y\_obs **-** theta[0]**\***x0 **-** theta[1]**\***x1) **\*** x0)     dth1 = np.mean(-2 **\*** (y\_obs **-** theta[0]**\***x0 **-** theta[1]**\***x1) **\*** x1)     **return**np.array([dth0, dth1])

: True

*You are correct! The answer “*True*” is correct because this is the correct Python function to compute gradient values.*

Gradients are a function of the entire dataset. : True

*You are correct! The answer “*True*” is correct because a gradient’s function is applied to the entire dataset to get the gradient value.*

Consider the following function call in Python:

mse\_gradient\_batch\_only(theta,batch\_indices,X,y\_obs)

What does the constructor “batch\_indices” represent? : The list of indices of data to calculate loss

*You are correct! The answer “*The list of indices of data to calculate loss*” is correct because this is what “*batch\_indices*” represents.*

The Python function np.split() is used to split an input array into a random number of subarrays. : False

*You are correct! The answer “*False*” is correct because the function*np.split()*is used to split an input array that is provided as a constructor to a function into a number of subarrays.*

Which of the following divides the entire dataset into a batch of datasets in order to save computing time? : Stochastic gradient descent

*You are correct! The answer “*Stochastic gradient descent*” is correct because stochastic gradient descent divides the entire dataset into a batch of datasets to save computing time.*

If the batch size is 1, the quality of the gradient is minimum, but the calculation is very fast. : True

*You are correct! The answer “*True*” is correct because the batch size is a parameter that gives the ability to tradeoff the quality of the gradient approximation against the runtime to compute the gradient approximation.*

In environments with large amounts of data, it is much more common to use mini-batch gradient descent. : True

*You are correct! The answer “*True*” is correct because the cost of computing the gradient on the entire dataset is too high, resulting in very slow algorithm training times.*

If the model is pushed to higher levels of complexity, thereby excessively increasing the parameters, the training error reduces to zero. : True

*You are correct! The answer “*True*” is correct because if the number of parameters is increased, the model is overparameterized, and there is an infinite number of valid choices of parameters that yield zero training errors.*

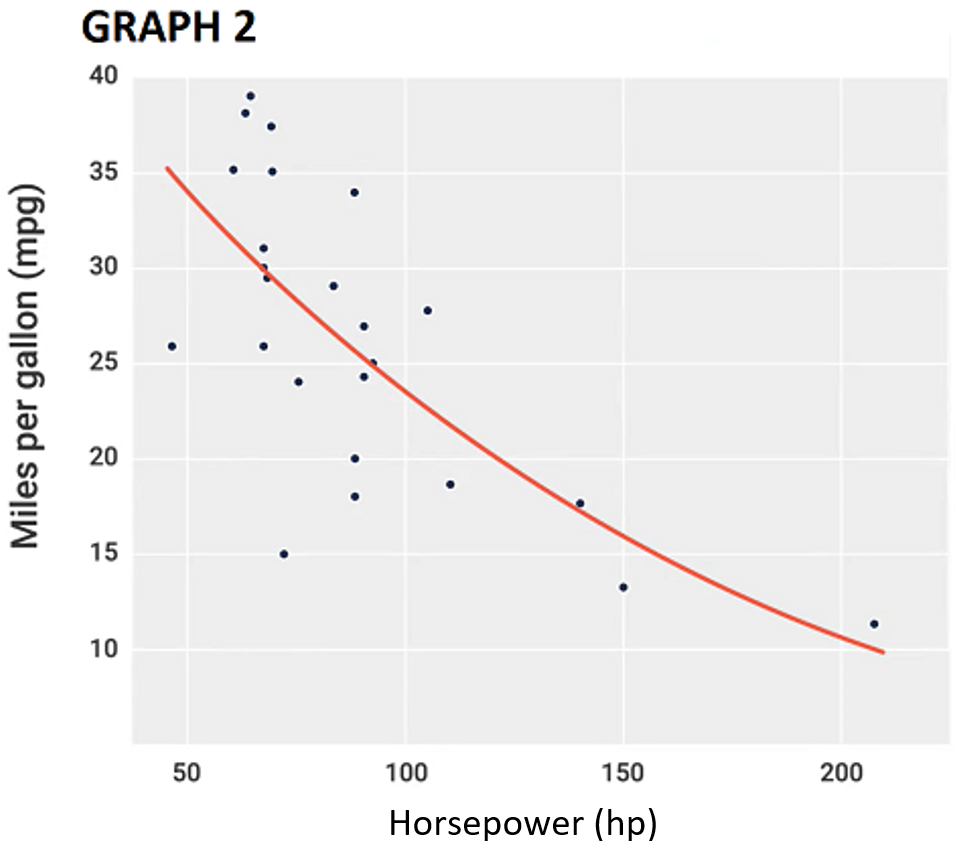
Overparameterized models that are trained with SGD act as if they are regularized. : True

*You are correct! The answer “*True*” is correct because overparameterized SGD results in a model that is implicitly regularized, yielding less wiggly behavior.*

Consider the following plots of two separate models:

Chart, scatter chart

Description automatically generated



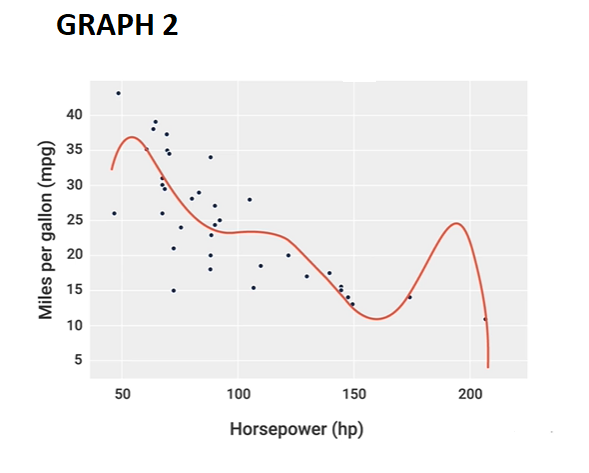
Which one has high bias? : Graph 1

*You are correct! The answer “*Graph 1*” is correct because the bias represents the fundamental inability of a model to fit the data, no matter what parameters are provided. Therefore, Graph 1 displays the inability to fit the model.*

Consider the following plots of two separate models:

Chart, scatter chart

Description automatically generated



Which one has high variance? : Graph 2

*You are correct! The answer “*Graph 2*” is correct because variance represents how sensitive the model is to the data. Therefore, it can be seen that Graph 2 is more sensitive to the data.*

As the model complexity increases, the bias will increase, and the variance will tend to decrease. : False

*You are correct! The answer “*False*” is correct because as the model complexity increases, the bias will decrease, and the variance will tend to increase.*

 ————— o —————

**Module 16**

**Notes:**

**Module Issues:**

**Quizes:**

 ————— o —————

**Module 17**

**Notes:**

**Module Issues:**

**Quizes:**