How do you ensure reproducibility when splitting data using train\_test\_split in Scikit-learn?

Group of answer choices



By using the same dataset each time.



By not shuffling the data before splitting.



By always using a 50-50 split for training and test.



By setting the random\_state parameter to a fixed number.

**Case:** As part of tuning your K-Nearest Neighbors (K-NN) classifier, you decide to use grid search to find the best value for 'n\_neighbors' among 3, 5, and 10. You have already initialized your K-NN classifier as knn = KNeighborsClassifier().  
**Question:** What is the correct Python code to set up this grid search using Scikit-learn?

Group of answer choices



param\_grid = {'k': [3, 5, 10]} grid\_search = GridSearchCV(knn, param\_grid)



param\_grid = {'n\_neighbors': [3, 5, 10]} grid\_search = GridSearchCV(knn, param\_grid)



knn = KNeighborsClassifier(n\_neighbors=[3, 5, 10]) grid\_search = GridSearchCV(knn)



knn = KNN() param\_grid = {'n\_neighbors': [3, 5, 10]} grid\_search = GridSearch(knn, param\_grid)

You are trying to read a CSV file named 'data.csv' and split it into features (X) and target (y). There is an error in the following code. What is the correct way to read the data and split it into X and y?

import pandas as pd

# Incorrect code  
df = pd.readcsv('data.csv')  
X = df.drop('target', axis=1)  
y = df['target']

Group of answer choices



Replace df['target'] with df.drop('target', axis=1)



Replace pd.readcsv with pd.read\_csv



Replace df.drop('target', axis=1) with df['target']



Replace pd.readcsv('data.csv') with pd.read\_excel('data.csv')

**Case:** You are working on a machine learning project and have received a dataset in the form of a CSV file named 'project\_data.csv'. The dataset includes various features and a target variable.  
**Question:** What Python code would you write using Pandas to load this CSV file into a DataFrame for further analysis?

Group of answer choices



df = pd.DataFrame('project\_data.csv')



df = pd.read\_excel('project\_data.csv')



df = pd.open('project\_data.csv')



df = pd.read\_csv('project\_data.csv')

What is the primary benefit of using k-fold cross-validation instead of a single validation set?

Group of answer choices



It is faster to compute



It uses every data point for both training and validation



It only needs a small dataset to be effective



It requires less memory

In the context of grid search, what does a 'grid' refer to?

Group of answer choices



A set of parameters and their possible values for tuning a model



The layout of data points in a dataset



The structure of a neural network



The partitioning of data in k-fold cross-validation

There is an error in the following code, which is meant to split a dataset into training and test sets and perform a grid search. Identify the correct fix:

from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.neighbors import KNeighborsClassifier  
import pandas as pd

# Assume df is a pre-loaded DataFrame  
X\_train, X\_test = train\_test\_split(df, test\_size=0.2, random\_state=42)

# Setting up the grid search  
knn = KNeighborsClassifier()  
param\_grid = {'n\_neighbors': [3, 5, 7]}  
grid\_search = GridSearchCV(knn, param\_grid)  
grid\_search.fit(X\_train, y\_train)

Group of answer choices



Replace X\_train, X\_test = train\_test\_split(df, test\_size=0.2, random\_state=42) with X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)



Replace knn = KNeighborsClassifier() with knn = KNeighborsRegressor()



Replace param\_grid = {'n\_neighbors': [3, 5, 7]} with param\_grid = {'k': [3, 5, 7]}



Replace grid\_search.fit(X\_train, y\_train) with grid\_search.fit(df)

In machine learning, what is the primary reason for performing a threefold split (training, validation, test) on the dataset?

Group of answer choices



To comply with standard data handling protocols.



To reduce the size of the training set for faster computation.



To allow for hyperparameter tuning without overfitting on the test set.



To use different algorithms on each split for comparison.

**Case:** You have a DataFrame df that contains feature columns and a separate Series y representing the target variable. You need to split this data into training and testing sets for your model, reserving 30% of the data for testing.  
**Question:** Which line of code using Scikit-learn would correctly perform this split?

Group of answer choices



X\_train, X\_test, y\_train, y\_test = df.train\_test\_split(y, test\_size=0.3)



X\_train, X\_test, y\_train, y\_test = split(df, y, test\_size=0.3)



X\_train, X\_test, y\_train, y\_test = train\_test\_split(df, y, test\_size=0.3)



X\_train, y\_train,X\_test, y\_test = train\_test\_split(df, y, train\_size=0.7)





