Python Institute

PCED

Certified Entry-Level Data Analyst with Python

QUESTION & ANSWERS

QUESTION: 1

You are a data analyst who needs to collect large volumes of data from a RESTful API that has pagination implemented. Which Python code snippet would be the most effective way to gather all the available data?

Option A : import requests

```
page = 1
while True:
    response = requests.get(f"https://api.example.com/data?page={page}")
    if response.status_code != 200:
        break
    data = response.json()
    # Process data
    page += 1
```

Option B : import requests

```
response = requests.get("https://api.example.com/data")
data = response.json()
```

Option C : import requests

```
pages = [1, 2, 3, 4, 5]
for page in pages:
    response = requests.get(f"https://api.example.com/data?page={page}")
    data = response.json()
    # Process data
```

Option D : import requests

```
for page in range(1, 100):
    response = requests.get(f"https://api.example.com/data?page={page}")
    data = response.json()
    # Process data
```

Correct Answer: A

Explanation/Reference:

Correct answer:

```
1. import requests
2.
3. page = 1
4. while True:
5.     response = requests.get(f"https://api.example.com/data?page={page}")
6.     if response.status_code != 200:
7.         break
8.     data = response.json()
9.     # Process data
10.     page += 1
```

This method employs a while-loop that checks the response status code to determine when to stop collecting data, making it more dynamic and robust.

Incorrect answers:

```
1. import requests
2.
3. for page in range(1, 100):
4.     response = requests.get(f"https://api.example.com/data?page={page}")
5.     data = response.json()
6.     # Process data
```

Using a fixed range of pages could either miss some pages or try to access pages that do not exist, resulting in errors or incomplete data.

```
1. import requests
2.
```

3. response = requests.get("https://api.example.com/data")

4. data = response.json()

This will only get data from the first page, missing out on all the other paginated data.

1. import requests

2.

3. pages = [1, 2, 3, 4, 5]

4. for page in pages:

5. response = requests.get(f"https://api.example.com/data?page={page}")

6. data = response.json()

7. # Process data

Manually specifying the page numbers limits the flexibility of the data collection.

QUESTION: 2

While working with a dataset, you notice some outliers that are not errors but extreme values that are relevant to your analysis. How should you handle them for ensuring the reliability and accuracy of your Python data analysis?

Option A : Ignore the outliers as they are not errors

Option B : Apply a log transformation

Option C : Replace the outliers with median values

Option D : Remove the outliers

Correct Answer: B

Explanation/Reference:

Apply a log transformation -> Correct. Applying a log transformation can reduce the impact of outliers while preserving their existence in the data, thereby ensuring both reliability and accuracy.

Remove the outliers -> Incorrect. Removing the outliers would not be ideal since they are relevant to the analysis.

Replace the outliers with median values -> Incorrect. Replacing the outliers with median values would dilute the extremeness which is considered relevant for the study.

Ignore the outliers as they are not errors -> Incorrect. Ignoring the outliers may skew the results and might not give an accurate analysis.

QUESTION: 3

You have a Pandas DataFrame that contains missing values. Which of the following methods will replace all NaNs with zeros?

Option A : df.fillna(value=0)

Option B : df.replace(to_replace=np.nan, value=0)

Option C : df['column_name'] = 0

Option D : df.dropna(subset=['column_name'])

Correct Answer: A

Explanation/Reference:

df.fillna(value=0) -> Correct. This is the correct method to replace all NaN values with zeros in a DataFrame.

df['column_name'] = 0 -> Incorrect. This will set all values in the specified column to zero, but won't address missing values in other columns.

df.replace(to_replace=np.nan, value=0) -> Incorrect. This will also replace NaNs with zeros, but df.fillna() is more idiomatic for this specific purpose.

df.dropna(subset=['column_name']) -> Incorrect. This removes rows where the specified column has NaN, rather than replacing NaN with zero.

QUESTION: 4

Your Python script is taking significantly longer than expected to complete. Which Python standard library could provide you with granular insights into the time taken by different parts of your code?

Option A : os

Option B : cProfile

Option C : sys

Option D : pickle

Correct Answer: B

Explanation/Reference:

cProfile -> Correct. This library can profile Python code to give you a detailed breakdown of time spent in each function, helping you identify bottlenecks.

os -> Incorrect. The os library is used for interacting with the operating system and doesn't provide profiling functionalities.

sys -> Incorrect. The sys library is used for accessing Python interpreter variables and doesn't provide profiling functionalities.

pickle -> Incorrect. The pickle library is used for object serialization and doesn't help with code profiling.

QUESTION: 5

You are analyzing the relationship between hours of study and exam scores among students. After calculating, you find a correlation coefficient (r) of 0.9. Which of the following is the best interpretation?

Option A : Hours of study has a strong, positive correlation with exam scores

Option B : Hours of study has a moderate, negative effect on exam scores

Option C : Exam scores have no relationship with hours of study

Option D : Hours of study causes high exam scores

Correct Answer: A

Explanation/Reference:

Hours of study has a strong, positive correlation with exam scores -> Correct. A correlation coefficient of 0.9 is close to 1, indicating a strong positive correlation between hours of study and exam scores. However, correlation does not imply causation, so we cannot say that more hours of study causes higher exam scores, or that it's the only factor affecting exam scores.

QUESTION: 6

You are tasked with merging two datasets, df1 and df2, on a common column 'id'. df1 has a column 'value1' and df2 has a column 'value2'. After merging, you need to create a new column called 'total_value' that sums 'value1' and 'value2'. Which of the following code snippets accomplishes this?

Option A : df_merged = pd.concat([df1, df2], keys='id')

df_merged['total_value'] = df_merged['value1'] + df_merged['value2']

Option B : df_merged = df1.merge(df2)

df_merged['total_value'] = df1['value1'] + df2['value2']

Option C : df_merged = df1.join(df2, on='id')

df_merged['total_value'] = df1['value1'] + df2['value2']

Option D : df_merged = pd.merge(df1, df2, on='id')

df_merged['total_value'] = df_merged['value1'] + df_merged['value2']

Correct Answer: D

Explanation/Reference:

Correct answer:

df_merged = pd.merge(df1, df2, on='id')

df_merged['total_value'] = df_merged['value1'] + df_merged['value2']

This option correctly merges the DataFrames on the 'id' field and correctly sums the 'value1' and 'value2' columns.

Incorrect answers:

df_merged = df1.join(df2, on='id')

df_merged['total_value'] = df1['value1'] + df2['value2']

The join function would not properly merge on the 'id' field in this format, and the summing would be incorrect due to wrong source DataFrames.

df_merged = pd.concat([df1, df2], keys='id')

df_merged['total_value'] = df_merged['value1'] + df_merged['value2']

pd.concat is not the correct way to merge DataFrames based on a common column.

df_merged = df1.merge(df2)

df_merged['total_value'] = df1['value1'] + df2['value2']

The merging would fail because it does not specify the column to merge on. Also, the summing refers to the wrong DataFrames.

QUESTION: 7

You have a linear regression model with 5 features. After checking the VIF (Variance Inflation Factor) for multicollinearity, one of the features has a VIF of 12. What should you consider doing?

Option A : Increase the sample size to improve the model.

Option B : Remove the feature with the high VIF from the model.

Option C : Use L1 regularization to mitigate the effect of multicollinearity.

Option D : Ignore it, as VIF of 12 is acceptable.

Correct Answer: B

Explanation/Reference:

Remove the feature with the high VIF from the model. -> Correct. Removing the feature with a high VIF is often the simplest way to deal with multicollinearity.

Ignore it, as VIF of 12 is acceptable. -> Incorrect. Generally, a VIF above 5-10 indicates a problematic amount of collinearity.

Increase the sample size to improve the model. -> Incorrect. Increasing the sample size won't necessarily solve the multicollinearity problem.

Use L1 regularization to mitigate the effect of multicollinearity. -> Incorrect. While regularization techniques like L1 (Lasso) can help in dealing with multicollinearity to some extent, they don't necessarily eliminate the root problem.

QUESTION: 8

You have a dataset with 15 features and you want to visualize it on a 2D plot for exploratory data analysis. What would be the first step in using PCA (Principal Component Analysis) to achieve this?

Option A : Perform clustering on the dataset

Option B : Apply a linear regression model to reduce dimensions

Option C : Use a scatter plot to visualize all the features

Option D : Standardize the dataset

Correct Answer: D

Explanation/Reference:

Standardize the dataset -> Correct. The first step in PCA is often to standardize the dataset so that each feature has a mean of 0 and a standard deviation of 1.

Perform clustering on the dataset -> Incorrect. Clustering is not the first step in PCA.

Use a scatter plot to visualize all the features -> Incorrect. A scatter plot is not useful for visualizing high-dimensional data directly.

Apply a linear regression model to reduce dimensions -> Incorrect. Linear regression is not used for dimensionality reduction in the context of PCA.

QUESTION: 9

You have a DataFrame df that contains duplicate rows. Your task is to remove duplicates based on two columns: 'A' and 'B'. What is the correct line of code to remove such duplicates while keeping the first occurrence?

Option A : df.drop_duplicates(subset=['A', 'B'], keep=False)

Option B : df.drop_duplicates()

Option C : df.unique(subset=['A', 'B'])

Option D : df.drop_duplicates(subset=['A', 'B'], keep='first')

Correct Answer: D

Explanation/Reference:

df.drop_duplicates(subset=['A', 'B'], keep='first') -> Correct. This is the correct approach.

df.drop_duplicates(subset=['A', 'B'], keep=False) -> Incorrect. It removes all duplicates, including the first occurrence.

df.drop_duplicates() -> Incorrect. It removes duplicate rows based on all columns.

df.unique(subset=['A', 'B']) -> Incorrect. It is incorrect because DataFrames don't have a .unique() method for this purpose.

QUESTION: 10

Which of the following libraries in Python is not commonly used for web scraping to collect data?

Option A : Scrapy

Option B : Selenium

Option C : Pandas

Option D : BeautifulSoup

Correct Answer: C

Explanation/Reference:

Pandas -> Correct. Although Pandas is a data manipulation library, it is not commonly used for web scraping.

BeautifulSoup -> Incorrect. BeautifulSoup is often used to parse HTML and XML documents, making it useful for web scraping.

Scrapy -> Incorrect. Scrapy is a popular framework used specifically for web scraping.

Selenium -> Incorrect. Selenium is used for web scraping, especially when JavaScript is involved or when you need to

simulate user interactions.

QUESTION: 11

Given a DataFrame df:

df = pd.DataFrame({

'A': [1, 2, 3],

'B': [4, 5, 6]

})

You want to square each individual element in the DataFrame. Which of the following code will accomplish this?

Option A : df.applymap(lambda x: x**2)

Option B : df['A'].apply(lambda x: x**2)

Option C : df.map(lambda x: x**2)

Option D : df.apply(lambda x: x**2, axis=1)

Correct Answer: A

Explanation/Reference:

df.applymap(lambda x: x**2) -> Correct. applymap is used for element-wise operations on a DataFrame, perfect for squaring each individual element.

df.apply(lambda x: x**2, axis=1) -> Incorrect. Using apply with axis=1 operates on each row, not on individual elements.

df['A'].apply(lambda x: x**2) -> Incorrect. This only squares the elements in column 'A'.

df.map(lambda x: x**2) -> Incorrect. DataFrames do not have a map method; it's a Series method.

QUESTION: 12

While evaluating a logistic regression model using a confusion matrix, you notice that the True Positive rate is significantly higher than the True Negative rate. What does this observation imply?

Option A : The model is better at identifying the positive class than the negative class

Option B : The model performs equally well for both classes

Option C : The model has a high level of bias

Option D : The model is overfitting

Correct Answer: A

Explanation/Reference:

The model is better at identifying the positive class than the negative class -> Correct. A higher True Positive rate indicates that the model is more successful in identifying instances of the positive class as compared to the negative class.

The model is overfitting -> Incorrect. Overfitting is generally diagnosed through high variance in cross-validation scores, not from the rates in a confusion matrix.

The model has a high level of bias -> Incorrect. A high level of bias typically leads to underfitting, not differences between True Positive and True Negative rates.

The model performs equally well for both classes -> Incorrect. The question explicitly states that the True Positive rate is significantly higher than the True Negative rate.

QUESTION: 13

When would you typically use the iloc attribute as opposed to the loc attribute in a pandas DataFrame?

Option A : When you want to apply a lambda function to each row

Option B : When you want to drop missing values from the DataFrame

Option C : When you want to access rows and columns by label

Option D : When you want to access rows and columns by integer-location

Correct Answer: D

Explanation/Reference:

When you want to access rows and columns by integer-location -> Correct. iloc is used for purely integer-location based indexing.

When you want to access rows and columns by label -> Incorrect. This is the primary use case for loc, not iloc.

When you want to drop missing values from the DataFrame -> Incorrect. This task is typically accomplished using methods like dropna.

When you want to apply a lambda function to each row -> Incorrect. Methods like apply or applymap are more commonly used for this purpose.

QUESTION: 14

You are evaluating a logistic regression model using the ROC curve. Which of the following is true regarding the Area Under the Curve (AUC)?

Option A : A model with an AUC of 0 performs better than one with an AUC of 1

Option B : A higher AUC indicates that the model has higher bias

Option C : A model with an AUC closer to 1 performs better than one closer to 0

Option D : A model with an AUC of 0.5 is an excellent model

Correct Answer: C

Explanation/Reference:

A model with an AUC closer to 1 performs better than one closer to 0 -> Correct. An AUC closer to 1 indicates better model performance, as it suggests that the model does a good job distinguishing between the positive and negative classes.

A model with an AUC of 0.5 is an excellent model -> Incorrect. An AUC of 0.5 suggests that the model is no better than random guessing.

A model with an AUC of 0 performs better than one with an AUC of 1 -> Incorrect. An AUC of 0 indicates a model that

performs extremely poorly.

A higher AUC indicates that the model has higher bias -> Incorrect. A higher AUC generally indicates better model performance, not higher bias.

QUESTION: 15

Your dataset consists of geographical data points for different species of birds. You want to visualize the distribution of these species on a geographical map. Which Python library would be most appropriate for this kind of visualization?

Option A : Plotly

Option B : Geopandas

Option C : Seaborn

Option D : Matplotlib

Correct Answer: B

Explanation/Reference:

Geopandas -> Correct. Specifically designed for working with geographical/spatial data and plotting it effectively.

Matplotlib -> Incorrect. While powerful for many types of visualizations, it doesn’t handle geographical data well.

Seaborn -> Incorrect. Built on top of Matplotlib, it also is not optimized for geographical data.

Plotly -> Incorrect. Offers some geographic plotting, but not as specialized as Geopandas for this specific task.

QUESTION: 16

When using Python's requests library to collect data from a website through HTTP requests, which status code indicates that the requested resource has been moved permanently?

Option A : 403

Option B : 404

Option C : 200

Option D : 301

Correct Answer: D

Explanation/Reference:

301 -> Correct. 301 Moved Permanently status code indicates that the requested resource has been permanently moved to a new URI.

200 -> Incorrect. 200 OK status code indicates a successful HTTP request, not that the resource has been moved.

403 -> Incorrect. 403 Forbidden status code indicates that the client does not have permission to access the requested resource.

404 -> Incorrect. 404 Not Found status code indicates that the server could not find the requested resource.

QUESTION: 17

You have a DataFrame named df that has three columns: 'A', 'B', 'C'. The DataFrame has 100 rows. Your task is to create a DataFrame where each column 'A', 'B', 'C' is transformed into 3 columns each: 'A_min', 'A_max', 'A_mean', 'B_min', 'B_max', 'B_mean', 'C_min', 'C_max', 'C_mean'. Which of the following code snippets accomplishes this task?

Option A : new_df = df.groupby(['A', 'B', 'C']).agg(['min', 'max', 'mean'])

Option B : new_df = df.apply(['min', 'max', 'mean'])

Option C : new_df = df.pivot(columns=['A', 'B', 'C'], values=['min', 'max', 'mean'])

Option D : new_df = df.aggregate(

```
    {
        'A': ['min', 'max', 'mean'],
        'B': ['min', 'max', 'mean'],
        'C': ['min', 'max', 'mean'],
    }
)
```

Correct Answer: D

Explanation/Reference:

Correct answer:

```
new_df = df.aggregate(
    {
        'A': ['min', 'max', 'mean'],
        'B': ['min', 'max', 'mean'],
        'C': ['min', 'max', 'mean'],
    }
)
```

The aggregate function allows you to perform multiple aggregations on columns and produces the desired output.

Incorrect answers:

```
new_df = df.groupby(['A', 'B', 'C']).agg(['min', 'max', 'mean'])
```

This would group the DataFrame by all the columns, which is not the requirement.

```
new_df = df.pivot(columns=['A', 'B', 'C'], values=['min', 'max', 'mean'])
```

The pivot method is used for reshaping or transforming data and not for applying aggregation functions.

```
new_df = df.apply(['min', 'max', 'mean'])
```

This would perform the aggregation, but the column names would not be formatted as required ('A_min', 'A_max', 'A_mean', etc.)

QUESTION: 18

What is the most appropriate technique to handle missing values for a categorical column in a dataset before fitting it to a model?

Option A : Remove Rows with Missing Values

Option B : Mean Imputation

Option C : Linear Regression Imputation

Option D : Mode Imputation

Correct Answer: D

Explanation/Reference:

Mode Imputation -> Correct. Mode imputation replaces missing values with the most frequently occurring value in the column. This is a standard practice for handling missing values in categorical columns.

Mean Imputation -> Incorrect. Mean imputation is generally not applicable to categorical variables.

Remove Rows with Missing Values -> Incorrect. Removing rows could result in significant data loss and potentially introduce bias.

Linear Regression Imputation -> Incorrect. Linear regression imputation is typically used for numerical variables and would not be appropriate for categorical variables.

QUESTION: 19

Which of the following techniques should be avoided to ensure that your Python data scripting code is easily understandable and follows best practices?

Option A : Using descriptive variable names

Option B : Refactoring repetitive code into reusable functions

Option C : Documenting functions with docstrings

Option D : Using single-letter variable names for main variables

Correct Answer: D

Explanation/Reference:

Using single-letter variable names for main variables -> Correct. This should be avoided, as it can make the code hard to understand and maintain.

Using descriptive variable names -> Incorrect. This is a best practice, as it makes the code self-explanatory.

Documenting functions with docstrings -> Incorrect. This is also a best practice, as it provides additional information about the functionâ€™s purpose, parameters, and return values.

Refactoring repetitive code into reusable functions -> Incorrect. This is considered a best practice, as it makes the code more modular and easier to maintain.

QUESTION: 20

Suppose you are working on a classification problem with an imbalanced dataset. Which of the following techniques would not be effective for addressing the issue?

Option A : Using accuracy as the evaluation metric

Option B : Using the Synthetic Minority Over-sampling Technique (SMOTE)

Option C : Using different evaluation metrics such as F1-score.

Option D : Under-sampling the majority class

Correct Answer: A

Explanation/Reference:

Using accuracy as the evaluation metric -> Correct. Using accuracy as the evaluation metric in an imbalanced dataset can give misleading results, as the model might simply predict the majority class for all inputs.

Using the Synthetic Minority Over-sampling Technique (SMOTE) -> Incorrect. SMOTE is a popular method for over-sampling the minority class to balance the dataset.

Using different evaluation metrics such as F1-score. -> Incorrect. F1-score is better suited for evaluating models trained on imbalanced datasets.

Under-sampling the majority class -> Incorrect. Under-sampling the majority class can also balance the dataset, although it may result in the loss of useful information.

QUESTION: 21

You are pulling data from various databases using Python to compile a large dataset for analysis. How would you ensure the data’s accuracy and reliability?

Option A : Use the most recent data available in all databases.

Option B : Write the Python script in a way that it runs faster to save computational resources.

Option C : Compare a random subset of data with source records to validate accuracy.

Option D : Perform exploratory data analysis (EDA) only after data cleaning is done.

Correct Answer: C

Explanation/Reference:

Compare a random subset of data with source records to validate accuracy. -> Correct. Comparing a random subset of data with source records allows you to verify that the data was pulled correctly, which aids in ensuring its accuracy and reliability.

Use the most recent data available in all databases. -> Incorrect. Using the most recent data does not guarantee that the data is accurate or reliable.

Perform exploratory data analysis (EDA) only after data cleaning is done. -> Incorrect. Performing EDA after data cleaning is a good practice, but it doesn't inherently ensure the initial data is accurate or reliable.

Write the Python script in a way that it runs faster to save computational resources. -> Incorrect. The speed of the Python script does not affect the accuracy or reliability of the data.

QUESTION: 22

What is the primary reason to use PCA (Principal Component Analysis) over t-SNE when visualizing high-dimensional data?

Option A : PCA is non-linear while t-SNE is linear

Option B : PCA is more computationally efficient for large datasets

Option C : PCA retains local structure while t-SNE focuses on global structure

Option D : PCA gives a more accurate representation of distances between clusters

Correct Answer: B

Explanation/Reference:

PCA is more computationally efficient for large datasets -> Correct. PCA is more computationally efficient and works faster, especially on large datasets.

PCA is non-linear while t-SNE is linear -> Incorrect. The opposite is true; PCA is linear while t-SNE is non-linear.

PCA retains local structure while t-SNE focuses on global structure -> Incorrect. Again, the opposite is true. t-SNE retains local structure while PCA focuses on maximizing variance, which is a global structure.

PCA gives a more accurate representation of distances between clusters -> Incorrect. t-SNE is generally better at preserving the distances between clusters and capturing the local structure.

QUESTION: 23

Which of the following NumPy methods would you use to apply a function element-wise on a 1-D array, while maintaining the shape of the original array?

Option A : np.reshape()

Option B : np.apply_along_axis()

Option C : np.ravel()

Option D : np.vectorize()

Correct Answer: D

Explanation/Reference:

np.vectorize() -> Correct. It is the correct answer as it applies a function element-wise on input arrays, maintaining the shape of the original array.

np.apply_along_axis() -> Incorrect. It is incorrect as it applies a function along the specified axis but does not necessarily maintain the original shape for a 1-D array.

np.reshape() -> Incorrect. It is incorrect because it only changes the shape of an array but doesn't apply any function element-wise.

np.ravel() -> Incorrect. It is incorrect because it flattens an array into a 1-D array but doesn't apply any function element-wise.

QUESTION: 24

You are working with a large dataset containing multi-dimensional features related to customer behavior. The dataset is too large and complex to analyze easily. What technique could be most effective for reducing the dataset's dimensionality while retaining most of its original variance?

Option A : Principal Component Analysis (PCA)

Option B : One-hot encoding

Option C : Linear Discriminant Analysis (LDA)

Option D : Max-Min Scaling

Correct Answer: A

Explanation/Reference:

Principal Component Analysis (PCA) -> Correct. Principal Component Analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It is commonly used to make data easy to explore and visualize by reducing its dimensionality. Unlike other techniques, it aims to maintain as much of the original data's variance as possible in the reduced representation.

QUESTION: 25

You have the following Pandas DataFrame with some missing values:

```
import pandas as pd
import numpy as np

df = pd.DataFrame(
    {"A": [1, np.nan, 3], "B": [4, 5, np.nan], "C": [7, 8, 9]}
)
```

How would you fill the missing values in column 'A' with the mean value of that column?

Option A : df['A'] = df['A'].fillna(df['C'].mean())

Option B : df['A'].fillna(df['A'].mean(), inplace=True)

Option C : df.fillna({'A': df['A'].mean()})

Option D : df['A'].apply(lambda x: x.fillna(x.mean()))

Correct Answer: B

Explanation/Reference:

df['A'].fillna(df['A'].mean(), inplace=True) -> Correct. This fills the missing values in column 'A' with its mean and modifies the DataFrame in place.

df.fillna({'A': df['A'].mean()}) -> Incorrect. This returns a new DataFrame with the changes, but doesn't modify the original DataFrame in place.

df['A'].apply(lambda x: x.fillna(x.mean())) -> Incorrect. apply function is not the right approach here.

df['A'] = df['A'].fillna(df['C'].mean()) -> Incorrect. This fills the missing values in column 'A' with the mean of column 'C', not 'A'.

QUESTION: 26

You are working with a logistic regression model for a binary classification problem. You notice that the training error is significantly lower than the validation error. What could you do to improve the model's performance on the validation set?

Option A : Increase the model's learning rate.

Option B : Add more features to the model.

Option C : Remove all features and retrain the model.

Option D : Regularize the model by introducing L1 or L2 penalties.

Correct Answer: D

Explanation/Reference:

Regularize the model by introducing L1 or L2 penalties. -> Correct. Regularization techniques like L1 or L2 penalties can help the model generalize better by constraining the model parameters.

Add more features to the model. -> Incorrect. Adding more features could exacerbate the overfitting problem.

Increase the model's learning rate. -> Incorrect. Increasing the learning rate could make the model unstable and is unlikely to help with overfitting.

Remove all features and retrain the model. -> Incorrect. Removing all features would make the model useless for prediction.

QUESTION: 27

You have a dataset of a retail company's sales transactions, which includes details like customer ID, transaction amount, transaction date, and store ID. You're tasked with summarizing monthly revenue for each store. What would be the most appropriate data aggregation technique?

Option A : Group the data by store ID and sum the transaction amounts daily, then take the average to get the monthly revenue.

Option B : Group the data by customer ID and store ID, then sum the transaction amounts and divide by the number of unique customers for each month.

Option C : Group the data by transaction date and calculate the daily revenue, then aggregate this to the monthly level without considering store ID.

Option D : Group the data by store ID and sum the transaction amounts for each month.

Correct Answer: D

Explanation/Reference:

Group the data by store ID and sum the transaction amounts for each month. -> Correct. The most direct way to summarize the monthly revenue for each store is to group the data by store ID and then sum the transaction amounts for each month. This approach directly answers the task at hand without introducing unnecessary complexities.

QUESTION: 28

You need to collect data from an Excel file with multiple sheets. One of the sheets is named "Sales" and another is named "Inventory". You want to load the "Sales" sheet into a DataFrame named df_sales and the "Inventory" sheet into a DataFrame named df_inventory. Which of the following code snippets accomplishes this?

Option A :
```
xls = pd.Excel('file.xlsx')

df_sales = xls.parse('Sales')

df_inventory = xls.parse('Inventory')
```
Option B :
```
df_sales = pd.read_excel('file.xlsx', sheet_name='Sales')

df_inventory = pd.read_excel('file.xlsx', sheet_name='Inventory')
```
Option C :
```
df_sales = pd.read_csv('file.xlsx', sheet_name='Sales')

df_inventory = pd.read_csv('file.xlsx', sheet_name='Inventory')
```
Option D :
```
df_sales = pd.read_excel('file.xlsx').parse('Sales')

df_inventory = pd.read_excel('file.xlsx').parse('Inventory')
```

Correct Answer: B

Explanation/Reference:

Correct answer:

df_sales = pd.read_excel('file.xlsx', sheet_name='Sales')

df_inventory = pd.read_excel('file.xlsx', sheet_name='Inventory')

It reads each sheet separately into its respective DataFrame.

Incorrect answers:

xls = pd.Excel('file.xlsx')

df_sales = xls.parse('Sales')

df_inventory = xls.parse('Inventory')

It should be pd.ExcelFile instead of pd.Excel.

df_sales = pd.read_excel('file.xlsx').parse('Sales')

df_inventory = pd.read_excel('file.xlsx').parse('Inventory')

Incorrect method chaining. read_excel() doesn't return an ExcelFile object.

df_sales = pd.read_csv('file.xlsx', sheet_name='Sales')

df_inventory = pd.read_csv('file.xlsx', sheet_name='Inventory')

read_csv() is not used for reading Excel files.

QUESTION: 29

You are implementing an ETL pipeline to fetch data from a JSON-based API and load it into a data lake. The API provides a pagination feature that shows 100 records per page. Which of the following Python libraries is most suitable for performing HTTP requests to extract data from this API?

Option A : requests

Option B : os

Option C : numpy

Option D : pandas

Correct Answer: A

Explanation/Reference:

requests -> Correct. The requests library is specifically designed for making HTTP requests and is the most suitable for fetching data from an API.

numpy -> Incorrect. numpy is primarily for numerical computing and does not handle HTTP requests.

pandas -> Incorrect. pandas is great for data manipulation and analysis but is not tailored for making HTTP requests.

os -> Incorrect. The os library is for interacting with the operating system and doesn't handle HTTP requests.

QUESTION: 30

You are given a DataFrame df with a column 'Date' of type string in the format 'YYYY-MM-DD'. You want to filter the rows where the year is 2020. Which of the following code snippets is the most efficient way to do so?

Option A : df[pd.to_datetime(df['Date']).dt.year == 2020]

Option B : df[df['Date'].str.contains('2020')]

Option C : df[df['Date'].str.slice(0, 4) == '2020']

Option D : df[df['Date'].apply(lambda x: x.split('-')[0]) == '2020']

Correct Answer: C

Explanation/Reference:

df[df['Date'].str.slice(0, 4) == '2020'] -> Correct. It is the most efficient as it slices the string to compare directly, avoiding unnecessary conversions.

df[df['Date'].str.contains('2020')] -> Incorrect. It might include other dates where the string '2020' appears, which could be misleading.

df[pd.to_datetime(df['Date']).dt.year == 2020] -> Incorrect. It is correct but inefficient because it has to convert each date string to a Pandas Timestamp object.

df[df['Date'].apply(lambda x: x.split('-')[0]) == '2020'] -> Incorrect. It also works but is not as efficient because it involves splitting each string.

QUESTION: 31

You are working on a machine learning project where you need to build a model for image classification. The project also requires heavy numerical computations and data preprocessing. Which of the following Python libraries would be most suitable to complete your project successfully?

Option A : Seaborn, Statsmodels, OpenCV

Option B : Pandas, Matplotlib, Keras

Option C : Scikit-learn, Matplotlib, BeautifulSoup

Option D : NumPy, Scikit-image, TensorFlow

Correct Answer: D

Explanation/Reference:

NumPy, Scikit-image, TensorFlow -> Correct. It consists of NumPy for numerical computations, Scikit-image for image processing, and TensorFlow for building machine learning models, making it the most well-rounded option for the project.

Pandas, Matplotlib, Keras -> Incorrect. It includes Pandas and Matplotlib, which are excellent for data manipulation and visualization, respectively, but Keras alone may not be sufficient for heavy numerical computations.

Scikit-learn, Matplotlib, BeautifulSoup -> Incorrect. It contains Scikit-learn and Matplotlib, which are good for machine learning and visualization but lacks a dedicated library for image processing and heavy numerical computations.

Seaborn, Statsmodels, OpenCV -> Incorrect. It contains Seaborn and Statsmodels for visualization and statistical models but lacks libraries for machine learning and image processing.

QUESTION: 32

You are working with a dataset that includes both categorical and continuous features. Which of the following feature scaling techniques would be most appropriate for scaling only the continuous features?

Option A : Min-Max Scaling

Option B : One-Hot Encoding

Option C : Label Encoding

Option D : Ordinal Encoding

Correct Answer: A

Explanation/Reference:

Min-Max Scaling -> Correct. Min-Max Scaling is commonly used for scaling continuous features to a specific range, usually [0, 1].

One-Hot Encoding -> Incorrect. One-Hot Encoding is generally used for categorical features, not continuous features.

Label Encoding -> Incorrect. Label Encoding is generally used for categorical features and not recommended for continuous features.

Ordinal Encoding -> Incorrect. Ordinal Encoding is typically used for ordinal categorical features, not continuous ones.

QUESTION: 33

When creating a dashboard using Tableau, which option allows you to integrate real-time Python code execution into your dashboard?

Option A : Tableau Server Scripts

Option B : Tableau Webhooks

Option C : Tableau Extensions

Option D : Tableau Public

Option E : TabPy (Tableau Python Server)

Correct Answer: E

Explanation/Reference:

TabPy (Tableau Python Server) -> Correct. TabPy (Tableau Python Server) is an external service that allows the execution of Python code from within Tableau, and it can be utilized to bring real-time Python analytics into a Tableau dashboard. Tableau Extensions are more about adding additional features or visualizations to Tableau. Tableau Server Scripts and Webhooks are not primarily used for executing Python code. Tableau Public is a free service that allows you to publish Tableau dashboards, but it does not support real-time Python code execution.

QUESTION: 34

You are working on a project that requires real-time data collection from a WebSocket API. Which of the following Python libraries would be most appropriate for this task?

Option A : csv

Option B : websocket-client

Option C : urllib

Option D : scrapy

Correct Answer: B

Explanation/Reference:

websocket-client -> Correct. The websocket-client library in Python is designed to interact with WebSocket APIs and is suitable for real-time data collection tasks. Libraries like urllib and scrapy are generally more appropriate for HTTP-based data collection, and csv is for working with CSV files, not for real-time WebSocket interaction.

QUESTION: 35

You are working with a dataset that has a mix of numerical and textual data. Your job is to synthesize this information into a more digestible form. What technique could you use to handle both types of data effectively for downstream analysis?

Option A : Run a Principal Component Analysis (PCA) on the entire dataset.

Option B : Implement a Decision Tree algorithm to identify the most important features.

Option C : Use k-Nearest Neighbors (k-NN) to classify data points.

Option D : Utilize Natural Language Processing (NLP) for textual data and Min-Max scaling for numerical data.

Correct Answer: D

Explanation/Reference:

Utilize Natural Language Processing (NLP) for textual data and Min-Max scaling for numerical data. -> Correct. Since the dataset includes both numerical and textual data, a single technique is unlikely to be sufficient. Natural Language Processing (NLP) methods can be applied to textual data to extract meaningful features. Simultaneously, Min-Max scaling can be used to normalize the numerical data. This dual approach allows for more effective downstream analysis and synthesis of the information.

QUESTION: 36

You have developed a logistic regression model for a classification task. After training, you notice that both the training and validation errors are high. Which of the following might be the problem with your model?

Option A : The learning rate is too high.

Option B : The model is overfitting the data.

Option C : The model is too complex.

Option D : The model is underfitting the data.

Correct Answer: D

Explanation/Reference:

The model is underfitting the data. -> Correct. Underfitting occurs when both training and validation errors are high because the model is too simple to capture the underlying pattern.

The model is too complex. -> Incorrect. A too complex model typically overfits, leading to low training errors but high validation errors.

The learning rate is too high. -> Incorrect. A high learning rate would not generally result in high errors for both training and validation data.

The model is overfitting the data. -> Incorrect. Overfitting would result in low training errors but high validation errors.

QUESTION: 37

You are given a JSON file containing social data about users' interactions on a website. The JSON object includes an array named likes that should contain integers representing the number of likes each post has received. Which of the following Python code snippets would be most appropriate to validate that all elements in the likes array are indeed integers?

Option A : import json

```
data = json.load(open('data.json'))

if len(data['likes']) > 0:
    print("Data is valid")
else:
```

```
    print("Data is invalid")
```

Option B : import json

```
data = json.load(open('data.json'))

if all(isinstance(x, int) for x in data['likes']):
    print("Data is valid")
else:
    print("Data is invalid")
```

Option C : import json

```
data = json.loads('data.json')

if isinstance(data['likes'], list):
    print("Data is valid")
else:
    print("Data is invalid")
```

Option D : import json

```
data = json.load(open('data.json'))

if 'likes' in data.keys():
    print("Data is valid")
else:
    print("Data is invalid")
```

Correct Answer: B

Explanation/Reference:

Correct answer:

import json

```
data = json.load(open('data.json'))


if all(isinstance(x, int) for x in data['likes']):
    print("Data is valid")
else:
    print("Data is invalid")
```

It uses Python's all function to check if every element in the likes array is an integer, ensuring data validity. The other options fail to validate that each element in likes is an integer.

QUESTION: 38

While transforming extracted data to fit operational needs, which of the following operations is least likely to be performed?

Option A : Data enrichment

Option B : Data normalization

Option C : Data archiving

Option D : Data type conversion

Option E : Data cleansing

Correct Answer: C

Explanation/Reference:

Data archiving -> Correct. Data archiving is generally not part of the "Transform" stage of the ETL process. The "Transform" stage is focused on preparing data for operational needs or analytical queries. This usually includes operations like data cleansing, normalization, type conversion, and enrichment, but not archiving, which is typically a separate concern handled after data has been loaded and operationalized.