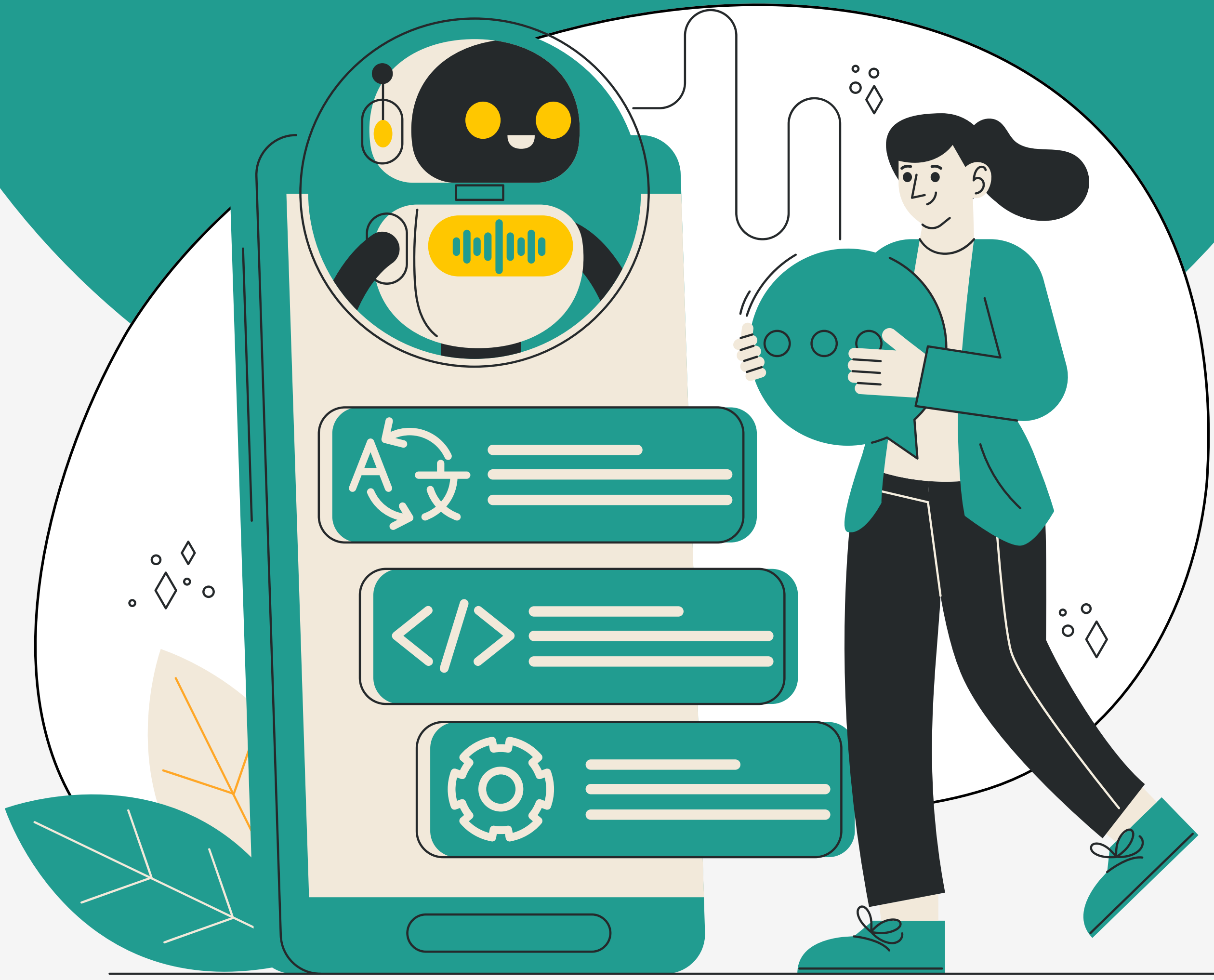
Feature engineering Interview Questions

(Practice Project)



PW SKILLS

Scenario-Based Questions (Easy/Medium/Hard)

1. Scenario: (Easy)
You are given a dataset with highly skewed numerical features. How would you handle this skewness before applying a machine learning model?

Answer:
You can apply a transformation to the skewed features, such as a logarithmic, square root, or Box-Cox transformation, to reduce the skewness and make the distribution more normal-like. This helps in improving the performance of many machine learning models that assume normality in the features.

2. Scenario: (Easy)
You have a dataset containing both numerical and categorical features. Explain how you would preprocess this dataset for a linear regression model.

Answer:
For a linear regression model, numerical features should be standardized or normalized, while categorical features should be one-hot encoded. Standardization ensures that all numerical features contribute equally to the model, and one-hot encoding allows categorical data to be represented in a format that the model can understand.

3. Scenario: (Medium)
You have been provided a dataset with missing values in some columns. Describe two different strategies to handle these missing values and when you would use each.

Answer:

1. Mean/Median Imputation: Replace missing values with the mean (for normally distributed data) or median (for skewed data) of the column. This is useful when the missing data is random and does not follow any pattern.

2. Using a Predictive Model: Train a model to predict the missing values based on other features. This approach is more sophisticated and can be used when there is a strong relationship between the missing values and other features.

4. Scenario: (Medium)
You are working with time-series data and need to perform feature engineering. What steps would you take to create meaningful features from the time component?

Answer:
You can extract features such as the day of the week, month, year, hour, or even holiday indicators from the timestamp. Additionally, you can create lag features (e.g., previous day's sales) and rolling statistics (e.g., rolling average of the last 7 days) to capture trends and seasonality in the data.

5. Scenario: (Medium)
You have a dataset with 100 features, some of which are highly correlated. How would you reduce the dimensionality of this dataset without losing significant information?

Answer:
You can apply Principal Component Analysis (PCA) to reduce the dimensionality. PCA transforms the original features into a set of linearly uncorrelated components ordered by the variance they explain. By retaining only the top components that explain most of the variance, you can reduce the number of features while preserving most of the information.

6. Scenario: (Medium)
In a classification problem, your dataset contains categorical variables with high cardinality. How would you handle these variables?

Answer:
For categorical variables with high cardinality, one-hot encoding may not be practical due to the large number of resulting features. Instead, you could use techniques like target encoding (replacing categories with the mean of the target variable within each category), frequency encoding, or embedding layers in neural networks for dimensionality reduction.

7. Scenario: (Hard)
You are tasked with building a model to predict customer churn. The dataset contains demographic information, transaction history, and customer interaction data. Describe your approach to feature engineering in this scenario.

Answer:
First, I would preprocess demographic data by handling missing values and encoding categorical variables. For transaction history, I would create features such as the total number of transactions, average transaction amount, and recency of the last transaction. Customer interaction data could be transformed into features like the frequency of interactions, sentiment analysis scores from text interactions, and time since the last interaction. Combining these features would provide a comprehensive view of customer behavior for the model.

8. Scenario: (Hard)
You are working on a dataset with numerical features on different scales. Explain why feature scaling is necessary and how you would implement it.

Answer:
Feature scaling is necessary because many machine learning algorithms, like SVMs and k-NN, are sensitive to the scale of the features. Without scaling, features with larger magnitudes could dominate the model's behavior. I would implement feature scaling using either Min-Max Scaling (scaling values to a range, typically 0 to 1) or Standardization (scaling values to have a mean of 0 and a standard deviation of 1).

9. Scenario: (Hard)
You are dealing with a dataset that has a large number of features (over 1,000). How would you determine which features to keep?

Answer:
I would start with feature selection techniques like Recursive Feature Elimination (RFE), which iteratively removes the least important features based on model performance. Additionally, I could use methods like L1 regularization (Lasso) to penalize and reduce less important features. Lastly, I would analyze feature importance scores from tree-based models like Random Forests or use techniques like PCA to reduce dimensionality.

10. Scenario: (Hard)
Imagine you are dealing with an imbalanced classification problem. What feature engineering techniques would you apply to handle the class imbalance?

Answer:
I would start by creating synthetic data for the minority class using techniques like SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN. Additionally, I could apply feature engineering techniques like adding class weights to the model, oversampling the minority class, or undersampling the majority class. I would also consider using ensemble methods like balanced Random Forest or XGBoost with class weights.

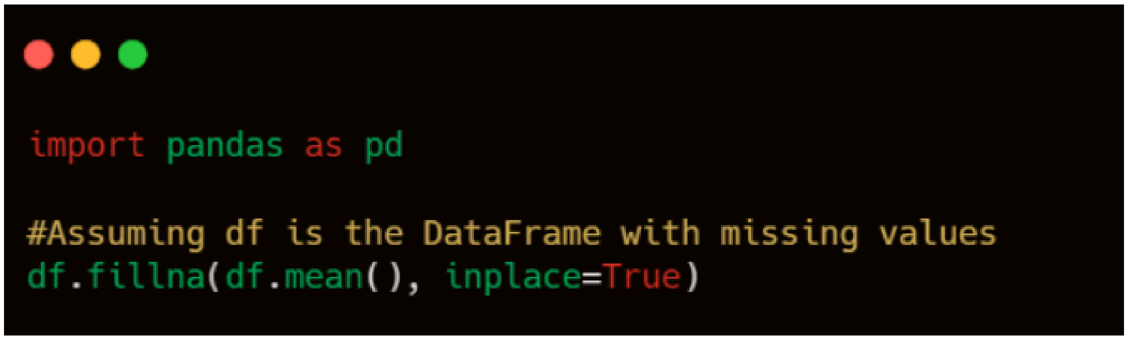
Practical-Based Questions (Easy/Medium/Hard)

1. Practical: (Easy)
Given a dataset with a categorical feature 'Color' having values ['Red', 'Blue', 'Green'], how would you preprocess this feature for a machine learning model?

Answer:
You can preprocess the 'Color' feature using one-hot encoding, which will convert the categorical values into separate binary columns: 'Color_Red', 'Color_Blue', and 'Color_Green'. Each column will contain 1 if the color matches the column name and 0 otherwise.

2. Practical: (Easy)
You are provided with a dataset where some numerical features have missing values. Write a Python code snippet to fill these missing values with the mean of their respective columns.

Answer:



```python
import pandas as pd

#Assuming df is the DataFrame with missing values
df.fillna(df.mean(), inplace=True)
```

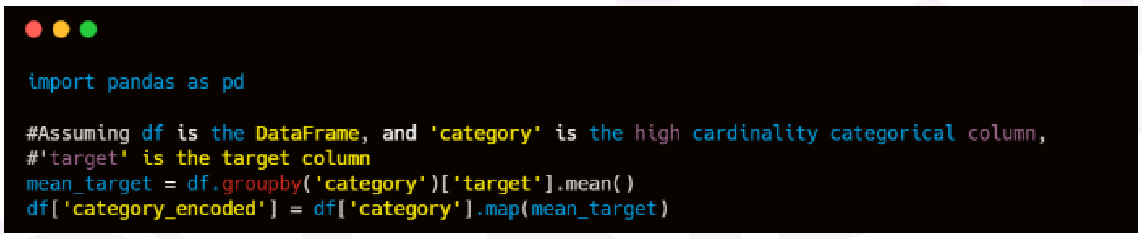
This code fills the missing values in the DataFrame `df` with the mean of each respective column.

3. Practical: (Medium)
Given a time-series dataset, how would you create a new feature that captures the rolling average of the past 7 days?

Answer:
```python
df['rolling_avg_7d'] = df['value_column'].rolling(window=7).mean()
```
This code creates a new feature `rolling_avg_7d` that contains the 7-day rolling average of the column `value_column` in the DataFrame `df`

4. Practical: (Medium)
You have a dataset with high cardinality categorical features. Implement a target encoding technique in Python.

Answer:

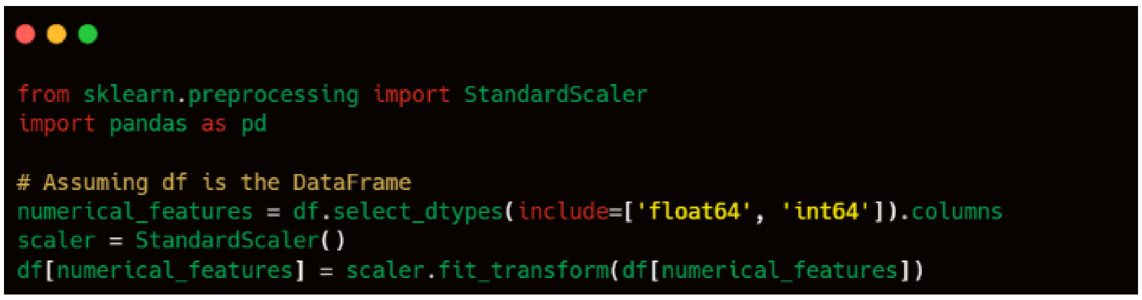


```
import pandas as pd

#Assuming df is the DataFrame, and 'category' is the high cardinality categorical column,
#'target' is the target column
mean_target = df.groupby('category')['target'].mean()
df['category_encoded'] = df['category'].map(mean_target)
```

This code replaces each category in the `category` column with the mean of the target variable for that category.

5. Practical: (Medium)
You are given a dataset with both categorical and numerical features. How would you apply StandardScaler to only the numerical features?

Answer:

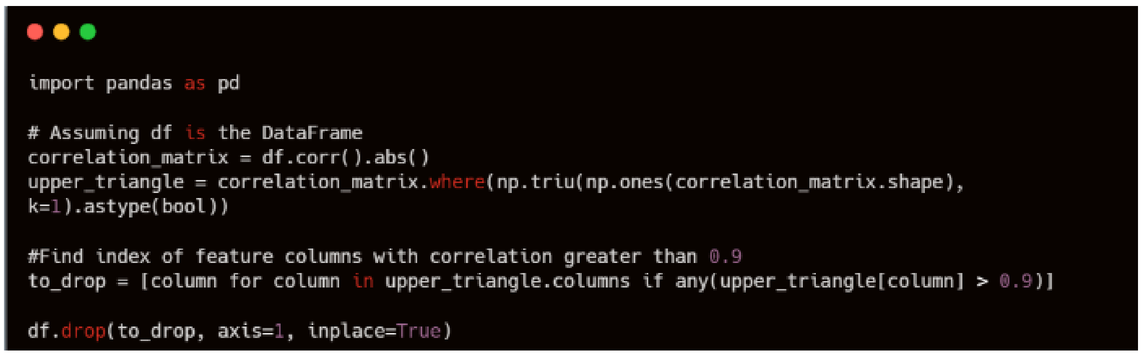


```
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Assuming df is the DataFrame
numerical_features = df.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()
df[numerical_features] = scaler.fit_transform(df[numerical_features])
```

This code applies `StandardScaler` only to the numerical features of the DataFrame `df`.

6. Practical: (Medium)
You have a dataset with highly correlated numerical features. Write a Python code snippet to remove one of each pair of features with a correlation higher than 0.9.

Answer:



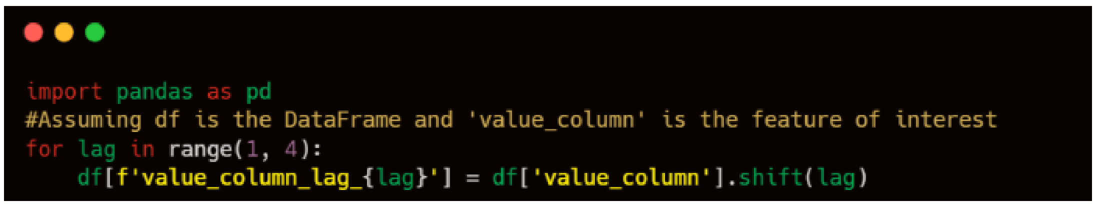
```
import pandas as pd

# Assuming df is the DataFrame
correlation_matrix = df.corr().abs()
upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))

#Find index of feature columns with correlation greater than 0.9
to_drop = [column for column in upper_triangle.columns if any(upper_triangle[column] > 0.9)]

df.drop(to_drop, axis=1, inplace=True)
```

\`\`\`

This code removes features with a correlation higher than 0.9 from the DataFrame `df`.

7. Practical: (Hard)
You have a dataset with time-series data. Write a Python code snippet to create lag features (e.g., previous day's value) for the last 3 days.

Answer:

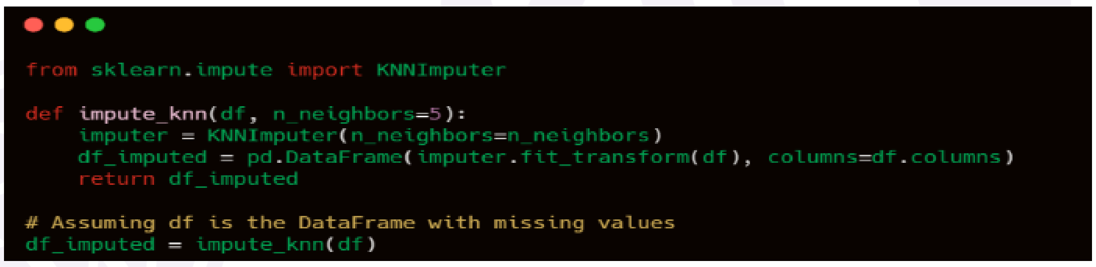


```python
import pandas as pd
#Assuming df is the DataFrame and 'value_column' is the feature of interest
for lag in range(1, 4):
    df[f'value_column_lag_{lag}'] = df['value_column'].shift(lag)
```

This code creates lag features for the previous 1, 2, and 3 days for the column value_column.

8. Practical: (Hard)
You are given a dataset with missing values. Write a Python function that applies K-Nearest Neighbors (KNN) imputation to handle the missing values.

Answer:

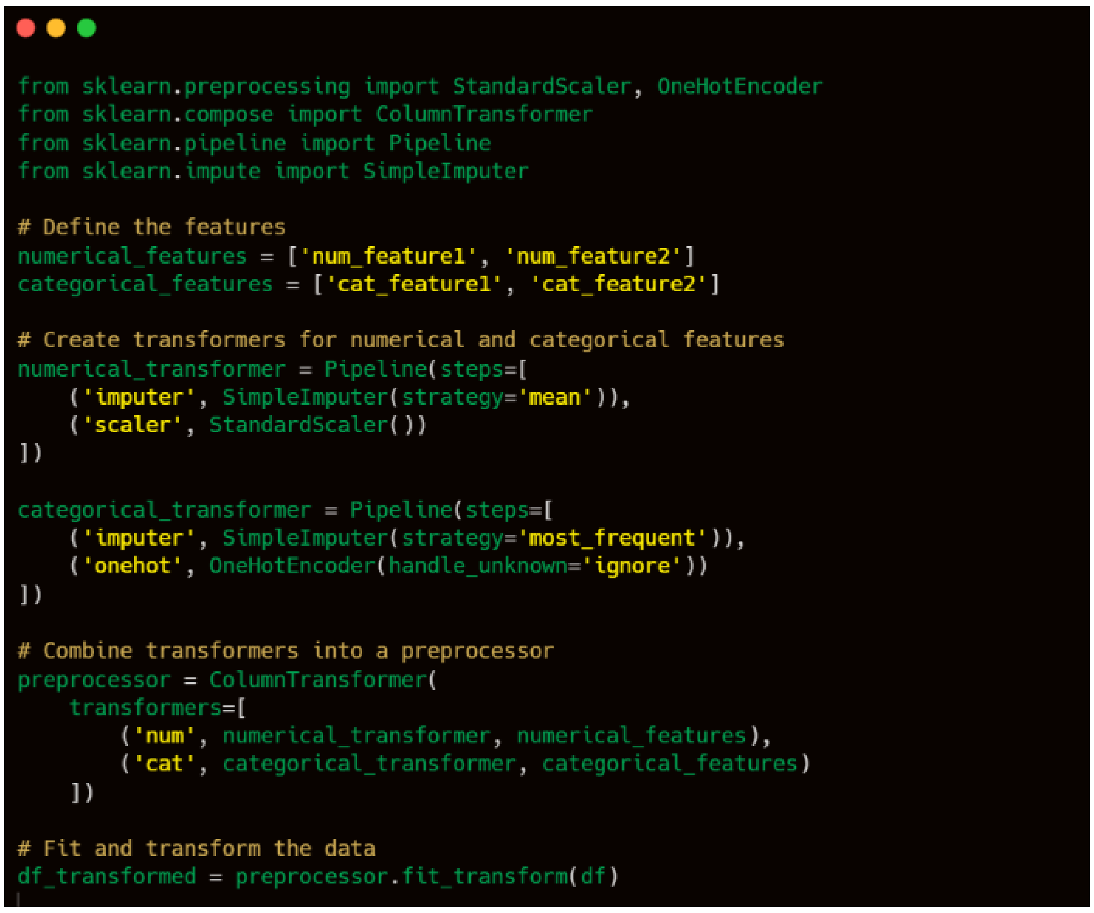


```python
from sklearn.impute import KNNImputer

def impute_knn(df, n_neighbors=5):
    imputer = KNNImputer(n_neighbors=n_neighbors)
    df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
    return df_imputed

# Assuming df is the DataFrame with missing values
df_imputed = impute_knn(df)
```

This function uses KNN imputation to handle missing values in the DataFrame df.

9. Practical: (Hard)
Given a dataset with mixed types of features (numerical, categorical), write a Python script to perform both one-hot encoding on categorical features and standardization on numerical features.

Answer:

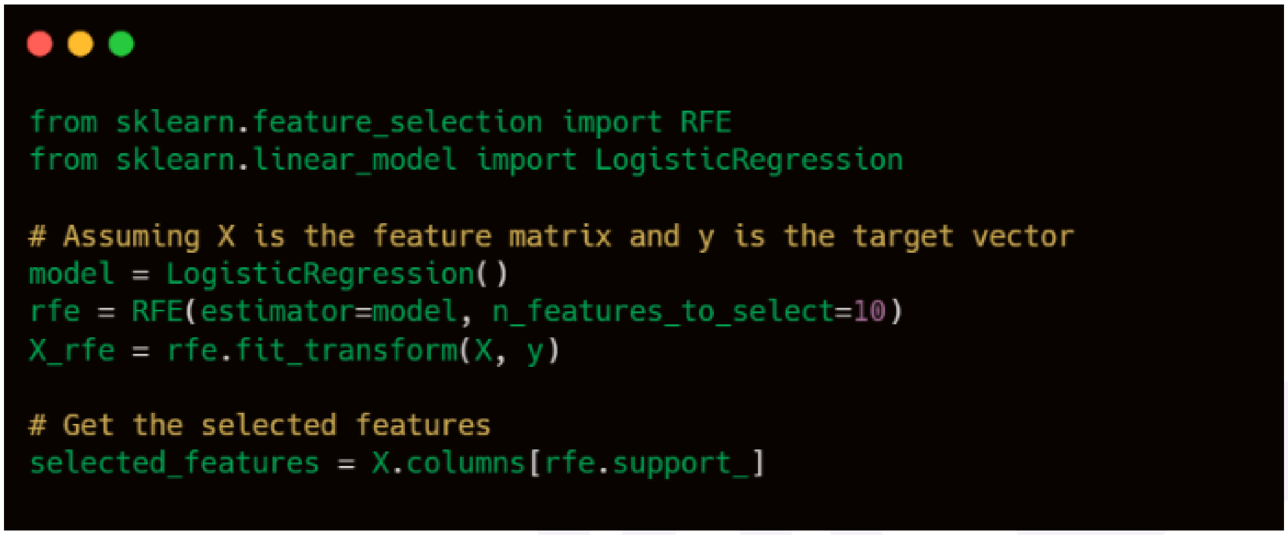


```python
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Define the features
numerical_features = ['num_feature1', 'num_feature2']
categorical_features = ['cat_feature1', 'cat_feature2']

# Create transformers for numerical and categorical features
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Combine transformers into a preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Fit and transform the data
df_transformed = preprocessor.fit_transform(df)
```

This script applies one-hot encoding to categorical features and standardization to numerical features using ColumnTransformer and pipelines.

10. Practical: (Hard)
You are given a dataset with various features and need to perform feature selection using Recursive Feature Elimination (RFE). Write a Python code snippet to perform this operation.

Answer:



```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

# Assuming X is the feature matrix and y is the target vector
model = LogisticRegression()
rfe = RFE(estimator=model, n_features_to_select=10)
X_rfe = rfe.fit_transform(X, y)

# Get the selected features
selected_features = X.columns[rfe.support_]
```

This code performs feature selection using RFE with a logistic regression model and selects the top 10 features.