■ Data Science Project - Exam Questions

Project: Machine Learning Data Science Application

Generated: 2025-10-27 11:32:44

Total Questions: 60+

1. Explain How Something Specific Works

A. Menu System & User Interface

Q1.1: Explain how the cross-platform keyboard input handling works in src/menu/key.py. What specific keys are mapped and how does the KeyHandler class normalize different operating systems?

Q1.2: Walk through the complete flow of how a user navigates through the menu system in src/menu/main.py. Explain the role of selected_option, isOptionSelected, and exiting variables.

Q1.3: Explain how the OptionsMenu class in src/menu/options.py handles different menu states. What happens when show_train_model_options is True vs False?

Q1.4: Describe the rendering system in src/menu/dataset.py. How does the render_two_column_layout method work and why is it designed this way?

B. Dataset Loading & Processing

Q1.5: Explain the complete dataset loading process in src/model/main.py's load_dataset method. What happens to the Excel file and how are train/test splits created?

Q1.6: Walk through the read_file method in src/model/main.py. Why does it read sheet 1 and sheet 2 specifically, and what does clean_columns do?

Q1.7: Explain how the get_basic_statistics method works. What statistics are calculated and how are they stored in the BasicStatistics dataclass?

Q1.8: Describe the split_dataset method. What columns are considered features vs target, and why is 'UNS' the target column?

C. Model Training & Evaluation

Q1.9: Explain the training process in src/model/train.py. How does the TrainModel class decide between KNN and Decision Tree training?

Q1.10: Walk through the evaluation process in src/model/evaluate.py. How are predictions generated and what metrics are calculated?

Q1.11: Explain how the load_additional_dataset_for_evaluation method works. When would a user want to evaluate on a different dataset?

Q1.12: Describe the predict method in src/model/evaluate.py. How does it handle new feature samples and what does it return?

D. Data Structures & Type Safety

Q1.13: Explain the Dataset dataclass in src/types/dataclass.py. What are all the fields and how do they relate to each other?

Q1.14: Walk through the to_dict method in the Dataset class. Why is this method needed and how does it handle serialization?

Q1.15: Explain the ModelClassifier and ModelEvaluation dataclasses. What's the difference between them and how are they used?

E. Utility Functions

Q1.16: Explain the Conversion class in src/utils/conversion.py. How does the recursive error handling work in the to method?

Q1.17: Describe the cross-platform input flushing in src/utils/sys.py. Why are there separate functions for Windows and Unix systems?

Q1.18: Explain the file operations in src/utils/file.py. How does get_file_path_by_name filter files and what's the purpose of the show_all parameter?

2. Change Things in the Code

A. Menu System Modifications

Q2.1: Modify the menu in src/config/main.py to add a new option 'View Model Performance History'. What changes would you need to make in the menu system?

Q2.2: Change the keyboard controls in src/menu/key.py to also accept 'j' and 'k' keys for navigation (like Vim). Show the exact code changes needed.

Q2.3: Modify src/menu/main.py to display the current dataset name in the menu title when a dataset is loaded. What variables and methods would you need to change?

Q2.4: Change the menu rendering in src/menu/dataset.py to show only 5 rows instead of 10 by default. Where would you make this change and what are the implications?

B. Model System Changes

Q2.5: Add a new model type 'Random Forest' to the training options. What files would you need to modify and what specific changes would you make?

Q2.6: Modify the KNN training in src/model/train.py to allow users to specify different distance metrics (euclidean, manhattan, etc.). Show the exact code changes.

Q2.7: Change the default number of neighbors in src/menu/options.py from 5 to 7. What other parts of the code might be affected by this change?

Q2.8: Modify the evaluation process to also calculate and display confusion matrix. What changes would you need in src/model/evaluate.py?

C. Data Processing Changes

Q2.9: Change the dataset loading to support CSV files in addition to Excel files. What modifications would you need in src/model/main.py and src/config/main.py?

Q2.10: Modify the feature selection to allow users to choose which columns to use as features instead of hardcoding ['STG', 'SCG', 'STR', 'LPR', 'PEG']. What changes would this require?

Q2.11: Change the train/test split to be configurable (e.g., 70/30 instead of using separate sheets). What modifications would you need to make?

Q2.12: Modify the data validation to check for outliers and display warnings. Where would you add this functionality?

3. Add Things to the Code

A. New Features

Q3.1: Add a feature to save model predictions to a CSV file. What new methods would you need to add and where would you place them?

Q3.2: Add a feature to compare two trained models side-by-side. What new classes or methods would you need to create?

Q3.3: Add a feature to visualize dataset distributions using matplotlib. What new dependencies and code would you need to add?

Q3.4: Add a feature to load previously saved progress from JSON files. What new menu options and methods would you need?

Q3.5: Add a feature to export evaluation results to a detailed report (HTML or PDF). What new classes would you need to create?

Q3.6: Add a feature to perform cross-validation on models. What modifications would you need in the training and evaluation process?

B. New Data Types

Q3.7: Add a new dataclass ModelComparison to store results when comparing two models. What fields would it need and how would it integrate with existing code?

Q3.8: Add a new dataclass PredictionHistory to track all predictions made during a session. How would this integrate with the existing Dataset class?

Q3.9: Add a new dataclass UserPreferences to store user settings like default model parameters. Where would you store and load these preferences?

C. New Utility Functions

Q3.10: Add a function to validate dataset quality (check for missing values, data types, etc.) before loading. Where would you place this and how would you integrate it?

Q3.11: Add a function to generate random test data for testing the application. What parameters would it need and how would you integrate it?

Q3.12: Add a function to backup all results to a timestamped folder. What file operations would you need and where would you call this function?

4. Remove Things (With/Without Breaking)

A. Safe Removals

Q4.1: Remove the flush_input functionality from src/utils/sys.py. What would break and what would continue to work? Explain the impact.

Q4.2: Remove the try_catch utility from src/utils/catch.py. What error handling would be lost and how would you replace it?

Q4.3: Remove the BasicStatistics calculation from dataset loading. What functionality would be lost and what would still work?

Q4.4: Remove the render_first_x_rows functionality from src/menu/dataset.py. What would users lose and what would still be available?

B. Breaking Removals

Q4.5: Remove the Model class from src/model/main.py. What would break immediately and what would be the cascade of failures?

Q4.6: Remove the Dataset dataclass from src/types/dataclass.py. What would break and why is this dataclass critical to the application?

Q4.7: Remove the KeyHandler class from src/menu/key.py. What would break and how would you fix the menu navigation?

Q4.8: Remove the Conversion class from src/utils/conversion.py. What input validation would be lost and how would you replace it?

C. Partial Removals

Q4.9: Remove only the Decision Tree model support. What files would you need to modify and what functionality would remain?

Q4.10: Remove only the KNN model support. What changes would you need to make and what would still work?

Q4.11: Remove the progress saving functionality. What would be lost and how would you modify the menu to remove this option?

Q4.12: Remove the cross-platform support (make it Windows-only). What code could you remove and what would break on other systems?

5. Fix Small Errors

A. Logic Errors

Q5.1: In src/menu/options.py line 52, there's a print statement that just prints 'n'. What is this supposed to do and how would you fix it?

Q5.2: In src/model/evaluate.py line 52, there's a print statement that prints 'n'. What should this be and how would you fix it?

Q5.3: The get_key method in src/menu/key.py returns the raw key as a fallback. What potential issues could this cause and how would you fix it?

Q5.4: In src/menu/main.py, the modulo operation for cycling through options could cause issues with negative numbers. How would you fix this?

B. Type Safety Issues

Q5.5: In src/types/dataclass.py, the to_dict method converts DataFrames to strings. What potential data loss could occur and how would you fix it?

Q5.6: The predict method in src/model/main.py could return None, but the calling code doesn't always handle this. How would you fix this?

Q5.7: In src/utils/file.py, the get_file_name function doesn't validate that the file exists before returning it. How would you fix this?

Q5.8: The Conversion class methods don't handle all possible input errors. What additional error handling would you add?

C. Performance Issues

Q5.9: The render_two_column_layout method in src/menu/dataset.py could be slow with large datasets. How would you optimize it?

Q5.10: The get_first_x_rows method loads all data into memory. How would you optimize this for large files?

Q5.11: The JSON serialization in Dataset.to_dict could be memory-intensive. How would you optimize it?

Q5.12: The menu re-rendering on every keypress could be inefficient. How would you optimize the rendering system?

■ Study Tips

• For each question, trace through the code step-by-step

• Understand the data flow from user input → processing → output

• Know the relationships between classes and modules

• Practice explaining the code out loud

• Test your understanding by making small changes and seeing what breaks

• Focus on the main patterns: MVC architecture, type safety, error handling, cross-platform compatibility

■ Project Structure Reference

project/ ■■■ main.py # Application entry point ■■■ datasets/ # Input datasets (Excel files) ■■■ results/ # Exported JSON files ■■■ src/ ■ ■■■ config/ # Configuration and constants ■ ■ ■■■ main.py # Menu options, file extensions, folder paths ■ ■■■ menu/ # User interface components ■ ■ ■■■ main.py # Main menu controller ■ ■ ■■■ options.py # Menu option handlers ■ ■ ■■■ dataset.py # Dataset visualization ■ ■ ■■■ key.py # Cross-platform keyboard input ■ ■■■ model/ # Machine learning logic ■ ■ ■■■ main.py # Core Model class ■ ■ ■■■ train.py # Model training implementations ■ ■ ■■■ evaluate.py # Model evaluation with metrics ■ ■■■ types/ # Data structures and type definitions ■ ■ ■■■ main.py # Generic type definitions ■ ■ ■■■ dataclass.py # Type-safe data structures ■ ■■■ utils/ # Utility functions and helpers ■ ■■■ catch.py # Error handling utilities ■ ■■■ conversion.py # Type conversion and input validation ■ ■■■ file.py # File I/O operations ■ ■■■ sys.py # System operations ■■■ README.md

Good luck with your exam! ■