ML DAY 6-7

09 June 2025 22:39

1. Project Objective

- Build a working **classification model from scratch**.
- Cover each stage: **data handling → feature processing → model training → evaluation**.

2. Data Pipeline

- **Data sourcing**: Use a straightforward toy dataset (commonly iris, titanic, or similar), showing how to load and inspect data.
- **Preprocessing**: Steps include handling missing values, encoding categorical variables, scaling, and splitting into training and testing sets.

3. Feature Engineering

- Derive new features or transform existing ones to enhance model performance.
- Demonstrates **one-hot encoding**, feature scaling, and possibly polynomial features or normalization.

4. Model Building

- Implement a classification algorithm—likely **Logistic Regression, Decision Tree, or k-Nearest Neighbors**.
- Explain algorithm selection rationale and theory in brief, followed by coding the model using a popular library (e.g., scikit-learn).

5. Training & Evaluation

- Train the model on the processed dataset.
- Evaluate with metrics such as **accuracy, precision, recall**, and visualize performance using **confusion matrix**.
- Possibly include cross-validation and discussion of overfitting vs underfitting.

6. Model Improvement & Iteration

- Suggest techniques to **tune hyperparameters**, add or remove features.
- Compare different models and iterate on preprocessing steps to improve performance.

7. Wrap-up & Learning Outcomes

- Reinforces the **ML workflow**: data ingestion → preparation → modeling → evaluation → iteration.
- Emphasizes reproducibility: clear code structure, modular functions, versioning, and documentation.

1. Understand the Business Context

- Emphasizes the necessity to **comprehensively understand the business goal** before diving into modeling
- Without clarity on the desired outcome, the project risks solving the wrong problem.

2. Transform Business Goal into ML Task

- Translate objectives into machine-learning tasks:
  - **Classification** (e.g., predict churn: yes/no)
  - **Regression** (e.g., forecast sales amount)
  - **Clustering**, **recommendation**, **anomaly detection**, etc.
- Choosing the correct ML task is foundational to pipeline design.

3. Define Input and Output

- Specify **target variable** (what you're predicting) and **features** (inputs).
- Consider data available at prediction time: features must be realistic and accessible when model runs.

4. Gather and Evaluate Data

- List all potential **data sources**.
- Assess data quality: are missing values, noise, or biases present?
- Gauge **quantity vs quality trade-offs**—sometimes a smaller high-quality dataset beats a large noisy one.

5. Set Measurable Evaluation Metrics

- Choose evaluation metrics aligned with business outcome:
  - **Accuracy**, **Precision/Recall**, **AUC** for classification.
  - **RMSE**, **MAE** for regression.
- Include **baseline performance**: e.g., always predicting the majority class.

6. Consider Constraints and Requirements

- Think about **operational constraints**:
  - Latency (speed of inference)
  - Resource usage (memory, CPU/GPU cost)
  - Regulatory or compliance factors
- These affect model architecture, deployment strategy, and monitoring.

7. Plan Model Iterations

- Create a roadmap for model development:
  1. **Baseline model** using basic techniques
  2. **Feature engineering iterations**
  3. **Model selection and tuning**
  4. **Testing and validation**
  5. **Deployment and monitoring**

1. Introduction to CSV Format

- Defines **CSV (Comma-Separated Values)** as a plain text format frequently used for tabular data exchange.
- Highlights its ubiquity: supported by tools like Excel, pandas, and most ML frameworks.

2. Loading CSV Data

- Illustrates how to **read CSV files** using pandas (pd.read_csv()).
- Covers important parameters:

    - sep: delimiter specification (comma, semicolon, tab, etc.)
    - header: row to use as column names
    - index_col: specify index column
    - usecols: load specific columns only.

3. Inspecting the Dataset

- Techniques to understand data:
    - df.head(n), df.tail(): preview records
    - df.info(), df.describe(): view schema and summary stats
    - Check for nulls with df.isnull().sum()

4. Handling Missing and Bad Data

- Demonstrates handling missing entries:
    - Remove rows with df.dropna()
    - Impute missing values (mean, median, constant) using fillna()
- Strategies for dealing with malformed lines—skip or raise errors depending on context.

5. Data Type Conversion

- Shows how to convert column typesDiscusses potential pitfalls: parsing dates, converting strings to numeric, handling unexpected formats.

6. Filtering and Subsetting

- Uses boolean indexing and .loc[] for filtering:

  python
  CopyEdit
  df[df['age'] > 30]
  df.loc[df['gender']=='Male', ['name', 'salary']]
- Applies masks to select specific rows/columns by condition.

7. Saving Back to CSV

- Writes DataFrame to CSV:

  python
  CopyEdit
  df.to_csv('cleaned.csv', index=False)
- Discusses important parameters: sep, header, encoding, and choosing whether to include index.