W2D2 - In Class Exercise

Lucas Yuki Nishimoto

Exercise: Understanding the ML Lifecycle and the type of algorithm:

Objective: Students will classify different ML problems as **supervised or unsupervised learning** and outline a basic ML lifecycle for one chosen problem.

Part 1: Identify the Learning Type

Task: Below are four different machine learning problems related to **Vancouver**. For each, students must:

1. Identify whether it's a **supervised** or **unsupervised** learning problem.
2. Justify their answer with reasoning.
3. Mention what kind of **algorithm** might be used (classification, regression, clustering, etc.).

Problem Scenarios (Canada-Specific):

● **Predicting Traffic Congestion in Vancouver:** The city wants to forecast traffic congestion based on historical traffic data, weather conditions, and road closures.
Supervised - Classification
Probably comes with the label Congestion or Not Congestion and the algorithm needs to find the patterns to classify the new cases.

● **Recommending Tourist Attractions in Vancouver:** A tourism board wants to suggest places to visit based on a visitor's past interests and travel patterns.
Unsupervised - Clustering
Based on the previous interests the algorithm will make groups and will suggest similar places

● **Detecting Fraudulent EI (Employment Insurance) Claims in Canada:** The government wants to identify suspicious EI claims by analyzing historical claim data.
Supervised - Classification
Considering a better accuracy the dataset should come with labels "Fraudulent" or "Not Fraudulent". That way the algorithm can learn the patterns for future classifications.

●**Grouping Vancouver Housing Market Trends:** A real estate company wants to segment neighborhoods based on price trends, demand, and property types.
Unsupervised - Clustering
As it is said, the idea is creating groups of similar houses, so it's a clustering problem.

Part 2: Define the ML Lifecycle for One Problem

Task: Pick one of the above problems and outline a simple **ML lifecycle** for solving it. You should cover the following steps:

1. **Problem Understanding:** What is the goal?
   Identify fraudulent EI

2. **Data Collection:** What data would be needed? (e.g., weather data, traffic sensors, real estate data, etc.)
   To detect fraudulent Employment Insurance claims, the model would focus on a small set of high-impact features that directly reflect inconsistencies in claimant behavior. The most essential data includes:

A. **Reported Earnings During the EI Period**
   Compares declared income with external signals of work activity, helping identify unreported earnings.

B. **Hours Worked Before the Claim**
   Detects inflated or inconsistent hours used to qualify for EI eligibility.

C. **Claim Frequency and Past Claim History**
   Identifies repeated or unusually frequent EI applications, a common pattern in fraudulent behavior.

D. **Employment History Consistency**
   Checks if employer information, job roles, and employment dates align with verified records.

E. **Benefit Amount Requested**
   Flags claims requesting unusually high or disproportionate benefit values relative to the claimant's known income.

3. **Data Preprocessing:** What aspects of the data do you consider important?

- **Verify and standardize data types:** ensure that numerical variables (income, hours worked, benefit amount) and categorical variables (region, industry, claim reason) are stored in the correct format.

- **Handle missing values:** apply median imputation for numerical fields and mode imputation for categorical fields to maintain consistency and reduce bias.

- **Remove duplicate records:** eliminate repeated claims or duplicated rows that may artificially inflate patterns.

- **Detect and treat outliers carefully:** since some outliers may represent actual fraud, they should not be blindly removed; instead, flagging, capping, or analyzing them is recommended.
- **Encode categorical variables:** convert categorical features to numerical format using One-Hot Encoding or Label Encoding so models can interpret them.
- **Perform train–test split:** split the dataset into training and testing subsets before scaling to prevent data leakage.
- **Apply feature scaling:** fit the scaler (e.g., StandardScaler or MinMaxScaler) only on the training data, then transform both training and test sets to ensure balanced model performance.

4. **Model Selection:** What algorithm(s) would be suitable?
   Random Forest and XGBoost are strong choices for supervised fraud detection because they handle complex, non-linear patterns in claimant and income data. If labeled fraud cases are limited, anomaly-detection models like Isolation Forest can also be used to identify suspicious claims.

5. **Model Training & Evaluation:** How would model performance be measured?
   The model would be trained using a standard 70/30 or 80/20 train–test split. Because fraud datasets are highly imbalanced, performance should be evaluated using metrics such as precision, recall, and F1-score rather than accuracy, ensuring the model correctly identifies fraudulent claims without excessive false positives.

6. **Deployment & Monitoring:** How can the model be deployed and maintained?
   The model can be deployed as an API or integrated into the EI processing system to score new claims in real time. Continuous monitoring is essential, using periodic retraining with new data, tracking performance metrics, and detecting model drift to ensure that fraud patterns are consistently identified as they evolve.

Summary of Algorithms and Metrics used:

Classification Algorithms

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Gradient Boosting (XGBoost, LightGBM, CatBoost)

5. Support Vector Machine (SVM)
6. k-Nearest Neighbors (k-NN)
7. Naive Bayes
8. Neural Networks (Deep Learning)

Common Classification Metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC
- Log Loss

Regression Algorithms

1. Linear Regression
2. Ridge Regression
3. Lasso Regression
4. Decision Tree Regression
5. Random Forest Regression
6. Gradient Boosting Regression (XGBoost, LightGBM, CatBoost)
7. Support Vector Regression (SVR)
8. k-Nearest Neighbors Regression (k-NN)
9. Neural Networks (Deep Learning)

Common Regression Metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 (R-Squared)

Clustering Algorithms

1. k-Means Clustering
2. Hierarchical Clustering
3. DBSCAN (Density-Based Spatial Clustering)

Common Clustering Metrics:

- Silhouette Score

Submission Guideline:

Please submit a pdf file with your detailed answers to the classrooms.

Grading Rubric:

- Understanding between supervised and unsupervised learning with the justification.
- Details provided in the ML Lifecycle.
- Aspects considered while building the Lifecycle.