AIL7024 Projects

Project 1: Predicting Compressive Strength of Concrete

Description

Concrete is the most widely used material in civil engineering, and its compressive strength is a critical factor in assessing structural performance and durability. The compressive strength of concrete depends on several parameters such as the ingredients used, water content, curing age, and more.

The objective of this project is to build predictive models for estimating the compressive strength of concrete using the provided dataset, and whether a concrete mix is fit for use in Reinforced Cement Concrete (RCC) as per Indian Standard IS 456:2000. The emphasis is on both methodological rigour (data preprocessing, modelling, parameter tuning) and insightful reporting (explaining design choices, comparing models, and interpreting results).

Dataset

Concrete Compressive Strength (UCI Machine Learning Repository)
https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength

The actual compressive strength (MPa) for a given mixture under a specific age (days) was determined from laboratory tests. Data is provided in raw form. You may need to analyze and preprocess this data. The dataset description is given as follows:

Variable	**Unit**	**Description**
Cement	kg/m³	Amount of cement
Blast Furnace Slag	kg/m³	Amount of slag
Fly Ash	kg/m³	Amount of fly ash
Water	kg/m³	Amount of water
Superplasticizer	kg/m³	Amount of admixture
Coarse Aggregate	kg/m³	Amount of coarse aggregate
Fine Aggregate	kg/m³	Amount of fine aggregate
Age	Days	Age of sample
Concrete Strength (target)	MPa	Compressive strength of concrete

Modelling

This problem needs to be approached in two stages:

Stage I: Regression (Predicting Compressive Strength)

- Implement Linear Regression and Polynomial Regression from scratch.
- Experiment with multiple train/test splits (e.g., 70:30, 80:20, cross-validation).
- Carry out feature engineering to construct new features.
- Implement Ridge and Lasso Regularization.
- Tune hyperparameters using methods like grid search or randomized search.
- Compare with advanced models like SVR, Gradient Boosting (XGBoost, LightGBM).
- Train the models, evaluate them on the test split, and report results and a comparative analysis across all models.

Stage II: Classification (Predicting Suitability for Indian RCC)

Indian Standard IS 456:2000 specifies that for Reinforced Cement Concrete (RCC), the characteristic 28-day compressive strength must be ≥ 20 MPa.

- Create a new target variable in form of a binary label for each sample: 1 (Fit for Indian RCC), 0 (Not Fit). [Note: Even if strength ≥ 20 MPa, age must be ≥ 28 Days]
- Check the class distribution after assigning labels to all samples in the dataset.
- Implement Logistic Regression from scratch.
- Implement Ridge and Lasso Regularization.
- Use classification models like Decision Trees, Random Forests, Gradient Boosting, Support Vector Machine (SVM Classifier).
- Handle class imbalance (if any) with stratified sampling, resampling, or class-weighting techniques.
- Tune hyperparameters using methods like grid search or randomized search.
- Train the models, evlauate them on the test split, and report results and a comparative analysis across all models..

Evaluation

The performance of your regression models can be reported using standard metrics:

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

The performance of your classification models can be reported using standard metrics:

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

Submission

Each team must submit:

1. **Codebase** in a ZIP file or GitHub repository with:
    - Clean, modular, and reproducible code.
    - Instructions in a `README` file for running experiments.
2. **Technical Report (PDF)** with the following sections:
    - Introduction
    - Preprocessing
    - Regression
        i. Modelling
        ii. Experiments
        iii. Results
    - Classification
        i. Modelling
        ii. Experiments
        iii. Results
    - Conclusion
    - Contributions (mandatory if working in a pair)

Project 2: Predicting Taxi Trip Duration and Ride Acceptance

Description

In the domain of intelligent transportation systems, predicting travel times is vital for route optimization, fleet management, and improving passenger experience. For ride-hailing services (like Uber, Ola, Rapido), it is equally important to predict whether a trip request will be accepted by a driver - a decision influenced by distance, time of day, weather, and traffic.

This project involves two connected tasks: predicting taxi trip duration and predicting whether a ride request will be accepted or rejected. The emphasis is on rigorous machine learning methodology (data cleaning, preprocessing, modelling, hyperparameter tuning) as well as insightful reporting (feature engineering, model comparison, and interpreting results).

Dataset

NYC Taxi Trip Duration Dataset
https://www.kaggle.com/competitions/nyc-taxi-trip-duration/data

This dataset contains detailed information about taxi rides in NYC, including pick-up and drop-off times, locations, passenger counts, and trip durations. The dataset description is given below:

- id: Unique identifier for each trip.
- vendor_id: Code indicating the driver associated with the trip record.
- pickup_datetime: Date and time when the meter was engaged.
- dropoff_datetime: Date and time when the meter was disengaged.
- passenger_count: Number of passengers in the vehicle (driver-entered value).
- pickup_longitude: Longitude coordinate of pickup point.
- pickup_latitude: Latitude coordinate of pickup point.
- dropoff_longitude: Longitude coordinate of drop-off point.
- dropoff_latitude: Latitude coordinate of drop-off point.
- store_and_fwd_flag: Indicates whether the trip record was held in vehicle memory before being sent (Y = stored and forwarded, N = sent directly).
- trip_duration: Duration of the trip in seconds (target for regression).

Modelling

Stage I: Regression (Trip Duration Prediction)

- Implement Linear Regression and Polynomial Regression from scratch.
- Use different train/test splits (70:30, 80:20) and cross-validation.
- Perform feature engineering to construct new features, like, but not limited to:
    - Trip distance (use coordinates)
    - Day of Week
    - Hour of Day
    - Weekend Indicator
    - Rush Hour Indicator

- Implement Ridge and Lasso Regularization to prevent overfitting.
- Tune hyperparameters (e.g. degree of polynomial, regularization strength) via grid search or randomized search.
- Compare with advanced models like SVR, Gradient Boosting (XGBoost, LightGBM).
- Train the models, evaluate them on the test split, and report results and a comparative analysis across all models.

Stage II: Classification (Ride Acceptance Prediction)

In real ride-hailing systems, drivers may reject trips that are too short (low profit), too long or far away (inconvenient) and during high-traffic/rush-hour, or if the request is coming in at odd hours or from remote pickup zones.

- Create a binary target variable for ride acceptance (1 = accept, 0 = reject) assuming the given rules:
    - Rule A (Short Trips Rejected): If trip distance < 1 km → **reject**.
    - Rule B (Very Long Trips Rejected): If trip distance > 50 km → **reject**.
    - Rule C (Rush-Hour Rejections for Short Trips): If trip distance < 3 km and rush hour → **reject**.
    - Rule D (Sleep-Hour Rejections): If pickup hour between 0000–0500 → **reject**.
    - Otherwise → **accept**.
- Check the class distribution after assigning labels to all samples in the dataset.
- Implement Logistic Regression from scratch.
- Implement Ridge and Lasso Regularization.
- Use classification models like Decision Trees, Random Forests, Gradient Boosting, Support Vector Machine (SVM Classifier).
- Handle class imbalance (if any) with stratified sampling, resampling, or class-weighting techniques.
- Tune hyperparameters using methods like grid search or randomized search.
- Train the models, evlauate them on the test split, and report results and a comparative analysis across all models..

Evaluation

The performance of your regression models can be reported using standard metrics:

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

The performance of your classification models can be reported using standard metrics:

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

Submission

Each team must submit:

1. **Codebase** in a ZIP file or GitHub repository with:
    - Clean, modular, and reproducible code.
    - Instructions in a `README` file for running experiments.
2. **Technical Report (PDF)** with the following sections:
    - Introduction
    - Preprocessing
    - Regression
        i. Modelling
        ii. Experiments
        iii. Results
    - Classification
        i. Modelling
        ii. Experiments
        iii. Results
    - Conclusion
    - Contributions (mandatory if working in a pair)

Project 3: Predicting Wine Quality using k-Nearest Neighbours

Description

The quality of wine is determined by various chemical properties such as acidity, sugar content, pH, alcohol concentration, and sulfur dioxide levels. Predicting wine quality based on physicochemical tests is a valuable problem in the beverage industry, as it allows producers to maintain consistency and optimize quality.

The objective of this project is to build predictive models for wine quality using the k-Nearest Neighbours (k-NN) algorithm for both regression and classification tasks. The emphasis is on data preprocessing, feature engineering, and insightful interpretations.

Dataset

Wine Quality Dataset (UCI Machine Learning Repository)
https://archive.ics.uci.edu/dataset/186/wine+quality

This dataset is related to variants of Portuguese wine. The dataset describes the amount of various chemicals present in wine and their effect on its quality. The quality ratings are provided by human experts. The dataset description is as follows:

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol
- quality (target: score between 0-10, assigned by human experts)

Modelling

Stage I: Regression (Predicting Wine Quality Score)

- Implement k-NN Regression from scratch.
- Use different train/test splits (70:30, 80:20) and cross-validation.
- Experiment with different distance metrics:
    - Euclidean Distance
    - Manhattan Distance
    - Minkowski Distance (generalized form)
- Try both uniform weighting (all neighbours equal) and distance-weighted k-NN.
- Tune the hyperparameter k using grid search and cross-validation.
- Compare results with baseline models like Linear Regression, Ridge Regression, etc.

- Train the models, evaluate them on the test split, and report results and a comparative analysis across all models.

Stage II: Classification (Predicting Wine Quality as Good/Bad)

- Create a binary classification target variable:
  - Quality ≥ 7 → "Good Quality" (label: 1)
  - Quality < 7 → "Low Quality" (label: 0)
- Implement k-NN Classifier from scratch.
- Compare with standard classifiers: Logistic Regression, Decision Trees, Random Forests, Support Vector Machine (SVM Classifier).
- Handle class imbalance (if any) with stratified sampling, resampling, or class-weighting techniques.
- Tune hyperparameters using methods like grid search or randomized search.
- Train the models, evlauate them on the test split, and report results and a comparative analysis across all models..

Evaluation

The performance of your regression models can be reported using standard metrics:

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

The performance of your classification models can be reported using standard metrics:

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

Submission

Each team must submit:

1. **Codebase** in a ZIP file or GitHub repository with:
   - Clean, modular, and reproducible code.
   - Instructions in a `README` file for running experiments.
2. **Technical Report (PDF)** with the following sections:
   - Introduction
   - Preprocessing
   - Regression: Modelling, Experiments, Results
   - Classification: Modelling, Experiments, Results
   - Conclusion
   - Contributions (mandatory if working in a pair)

Project 4: Cuisine Prediction

Description

The objective of this assignment is to develop a machine learning model to predict the cuisine of a given recipe based on its list of ingredients. This task involves classifying the type of cuisine (e.g., Indian, Mexican, Moroccan, etc.) for each recipe in the dataset. You will need to create features from each data item and build a model to perform this task.

Dataset

In the dataset, we include the recipe ID, the type of cuisine, and the list of ingredients of each recipe (of variable length). The data is stored in JSON format at the following link: https://drive.google.com/drive/folders/1i0OCcSjrXXadtnd-Sro6IdNBic-b2LYU?usp=drive_link

An example of a recipe node in train.json:

```
{
        "id": 24717,
        "cuisine": "indian",
        "ingredients": [
                "turmeric",
                "vegetable stock",
                "tomatoes",
                "garam masala",
                "naan",
                "red lentils",
                "red chili peppers",
                "onions",
                "spinach",
                "sweet potatoes"
        ]
}
```

In the test file test.json, the format of a recipe is the same as train.json, only the cuisine type is removed, as it is the target variable you are going to predict.

Evaluation

For simplicity, the evaluation metric for this project is overall Accuracy:
Accuracy = (tp + tn) / (tp + tn + fp + fn)
Where:
tp: True Positives
tn: True Negatives
fp: False Positives
fn: False Negatives

Submission

You should create a validation split and test the model's performance, and then finally do the predictions on the test set. Prepare the final submission file in the required format and submit it for evaluation. You must submit a CSV file with exactly 10,000 entries plus a header row. The file should have exactly 2 columns:

- Id (sorted in any order)
- Category (must be a string from all possible cuisines)

Example:
Id,Category
15717,mexican
25265,moroccan
6935,indian
46557,korean
8678,greek