ML-Assignment -2

Name: Kuruba Buchannagari Shanmukh Sai

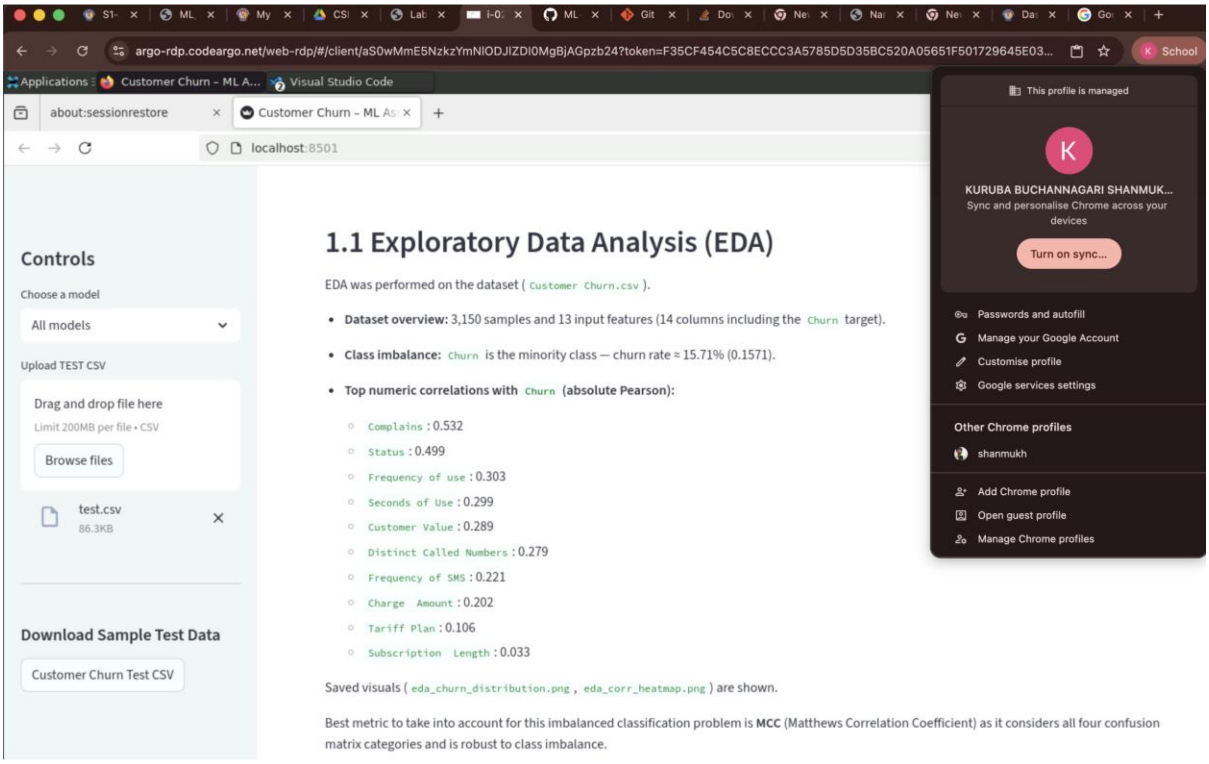
ID: 2025AB05092

Email: 2025AB05092@wilp.bits-pilani.ac.in

GitHub Link: https://github.com/Shanusai/ML-Assignment-2

Stream Lit app Link: https://ml-assignment-2-vqgo7ojusudlepj35tddfz.streamlit.app/

Bits Lab execution screenshot:



Choose a model: All models

Upload TEST CSV: test.csv 86.3KB

1.1 Exploratory Data Analysis (EDA)

EDA was performed on the dataset (`Customer Churn.csv`).

- **Dataset overview:** 3,150 samples and 13 input features (14 columns including the `Churn` target).
- **Class imbalance:** `Churn` is the minority class — churn rate ≈ 15.71% (0.1571).
- **Top numeric correlations with `Churn` (absolute Pearson):**
  - `Complains` : 0.532
  - `Status` : 0.499
  - `Frequency of use` : 0.303
  - `Seconds of Use` : 0.299
  - `Customer Value` : 0.289
  - `Distinct Called Numbers` : 0.279
  - `Frequency of SMS` : 0.221
  - `Charge  Amount` : 0.202
  - `Tariff Plan` : 0.106
  - `Subscription  Length` : 0.033

Saved visuals (`eda_churn_distribution.png` , `eda_corr_heatmap.png`) are shown.

Best metric to take into account for this imbalanced classification problem is **MCC** (Matthews Correlation Coefficient) as it considers all four confusion matrix categories and is robust to class imbalance.

Readme :

Assignment 2- Customer Churn Prediction

a. Problem Statement

Predict whether a customer will churn based on their usage and demographic data. The goal is to build and compare multiple machine learning models to identify the best approach for churn prediction.

b. Dataset Description [1 mark]

- **Source:** Customer Churn.csv
- **Target:** Churn (0 = No churn, 1 = Churn). Note: this is an imbalanced dataset — the churn class (1) is the minority class. Take care during model development and evaluation (use stratified splits, consider class weights or resampling, and prefer metrics such as AUC, precision, recall, F1 and MCC over accuracy alone).
- **Features:**
- **Features:** (each feature includes suggested type and brief notes)
  - Call Failure — numeric (count) or binary indicator: number of failed calls; higher values often indicate service issues and higher churn risk. Treat as numeric, cap outliers.
  - Complains — numeric (count) or binary: customer complaints logged; strong positive signal for churn. Consider time-windowed counts.
  - Subscription Length — numeric (months/days): tenure in service; longer tenure usually correlates with lower churn. Use as-is or bucket into bins.
  - Charge Amount — numeric (monetary): total/average charges; can indicate customer value or bill shock. Scale/normalize and examine skew.
  - Seconds of Use — numeric: total voice usage seconds in period; indicates engagement. Consider per-day averages.
  - Frequency of Use — numeric: number of calls/sessions in period; high usage typically reduces churn risk.
  - Frequency of SMS — numeric: number of SMS sent/received; another engagement signal.
  - Distinct Called Numbers — numeric: unique contacts called; higher diversity often means stronger network effects and lower churn.
  - Age Group — categorical (e.g., 18-25, 26-35): encode with one-hot or ordinal depending on model; may interact with usage patterns.
  - Tariff Plan — categorical: subscription/tariff type; important for pricing-related churn — use one-hot or target encoding for high-cardinality plans.
  - Status — categorical/binary: active/inactive/paused; direct indicator of customer relationship state.
  - Age — numeric: customer age; can be used directly or to derive Age Group if missing.
  - Customer Value — numeric (e.g., lifetime value or average revenue): key feature for prioritizing retention — scale and handle missing carefully.

- Churn (Target) — binary: 0 = No churn, 1 = Churn. This is the minority class (imbalanced dataset).

c. Models Used [6 marks]

Six ML models were trained and evaluated:

- Logistic Regression
- Decision Tree
- kNN
- Naive Bayes
- Random Forest (Ensemble)
- XGBoost (Ensemble)

Comparison Table: Evaluation Metrics

ML Model Name	Accuracy	AUC	Precision	Recall	F1	MCC
XGBoost	0.963	0.990	0.858	0.919	0.888	0.867
Random Forest	0.940	0.985	0.736	0.960	0.833	0.808
Decision Tree	0.944	0.912	0.802	0.859	0.829	0.797
kNN	0.941	0.961	0.772	0.889	0.826	0.794
Logistic Regression	0.871	0.926	0.562	0.818	0.667	0.606
Naive Bayes (Gaussian)	0.735	0.908	0.361	0.889	0.513	0.445

Observations on Model Performance [3 marks]

ML Model Name	Detailed Observation about model performance
Logistic Regression	Moderate overall (Accuracy 0.871, AUC 0.926). Good balance with recall (0.818) capturing most churns while precision (0.562) keeps false positives moderate; F1 (0.667) and MCC (0.606) make it a solid interpretable baseline.
Decision Tree	Strong and balanced (Accuracy 0.944, AUC 0.912) with good precision (0.802) and recall (0.859). F1 (0.829) and MCC (0.797) indicate reliable classification with interpretable decisions.

kNN	High AUC (0.961) and good recall (0.889); precision (0.772) is lower than top ensembles. F1 (0.826) and MCC (0.794) show kNN is competitive when features are scaled appropriately.
Naive Bayes (Gaussian)	Decent recall (0.889) but lower precision (0.361) and moderate accuracy (0.735). F1 (0.513) and MCC (0.445) indicate it's a fast baseline with limited overall effectiveness.
Random Forest	Very strong AUC (0.985) and excellent recall (0.960) with balanced precision (0.736). F1 (0.833) and MCC (0.808) show ensemble robustness and strong holdout performance.
XGBoost	Best overall (Accuracy 0.963, AUC 0.990). High precision (0.858) and recall (0.919) yield top F1 (0.888) and MCC (0.867), making XGBoost the recommended production model for performance-focused use.

Hyperparameter Trials

The script model/run_hyperparameter_search.py performs hyperparameter search for **each implemented model** using GridSearchCV (scoring = F1, CV = 5) and then:

- selects best hyperparameters
- evaluates best model on holdout split
- saves best model artifact (.joblib)
- saves summary CSV (hyperparameter_results.csv)

Example command

```
cd project-folder/model
python3 run_hyperparameter_search.py --data "../train.csv" --output artifacts
```

Run Streamlit App

```
cd project-folder
pip3 install -r requirements.txt
streamlit run streamlit_app.py
```