

Hack The Future 2025

Team Nexus

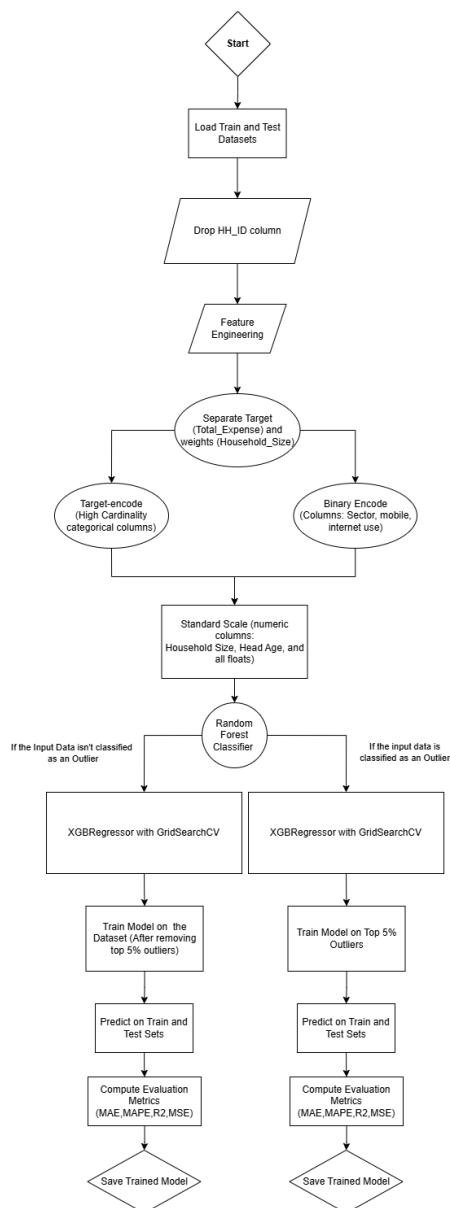
Github Link: https://github.com/SayuiGupta2005/Team_Nexus-HackTheFuture2025

User Guide

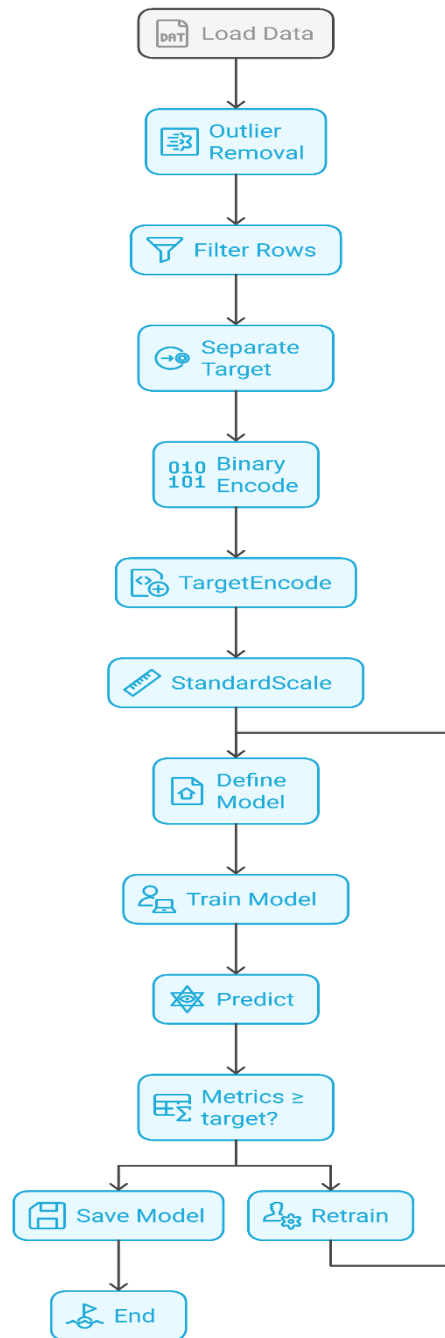
A) Hardware and Software requirements:

- Hardware needs: Basic laptop 4gb RAM
- Software needs: Python, VSCode, Nesstar

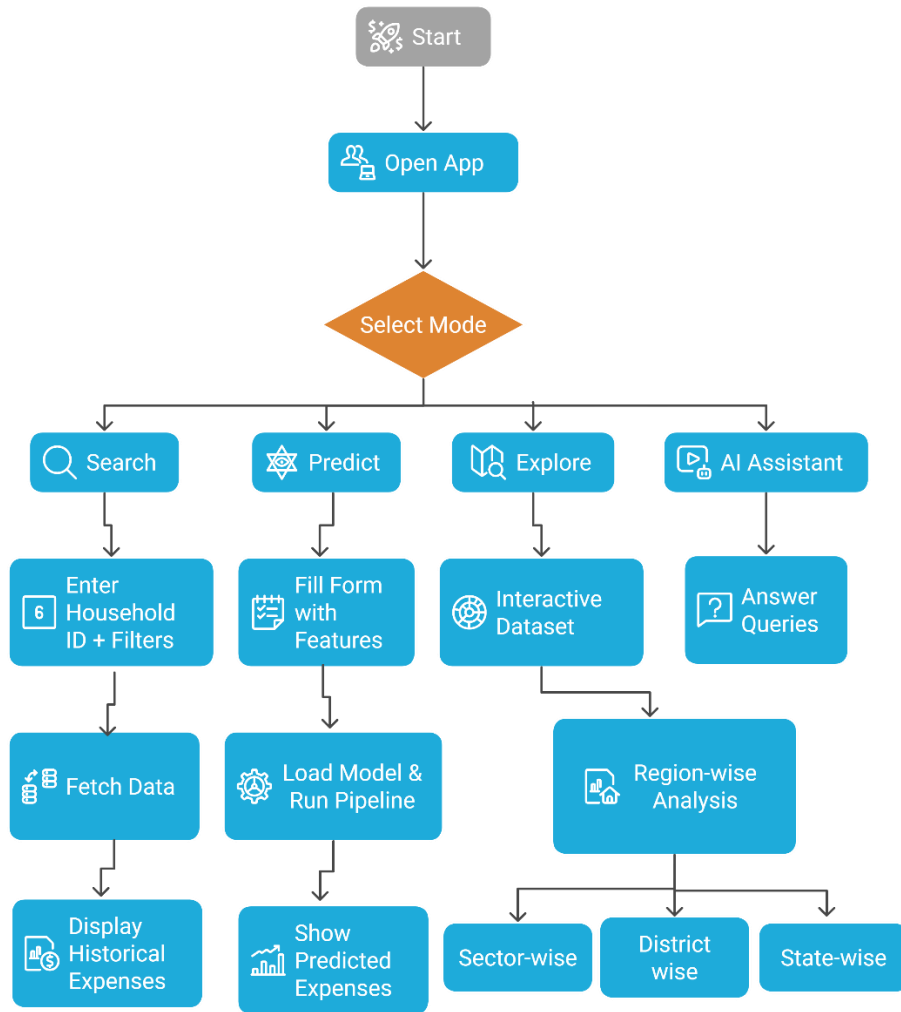
B) Approach:



Final XGBoost Regression Pipeline (v3)



Local Household Expense Web App User Journey



C) List of Python libraries used:

- pandas
- numpy
- scikit-learn
- category_encoders
- xgboost
- joblib
- matplotlib
- seaborn
- flask
- CORS

D) Environmental Details:

- Google Colab

- Kaggle Notebook

E) List of APIs Used:

- Groq – Open Source

F) List of files required to execute the program:

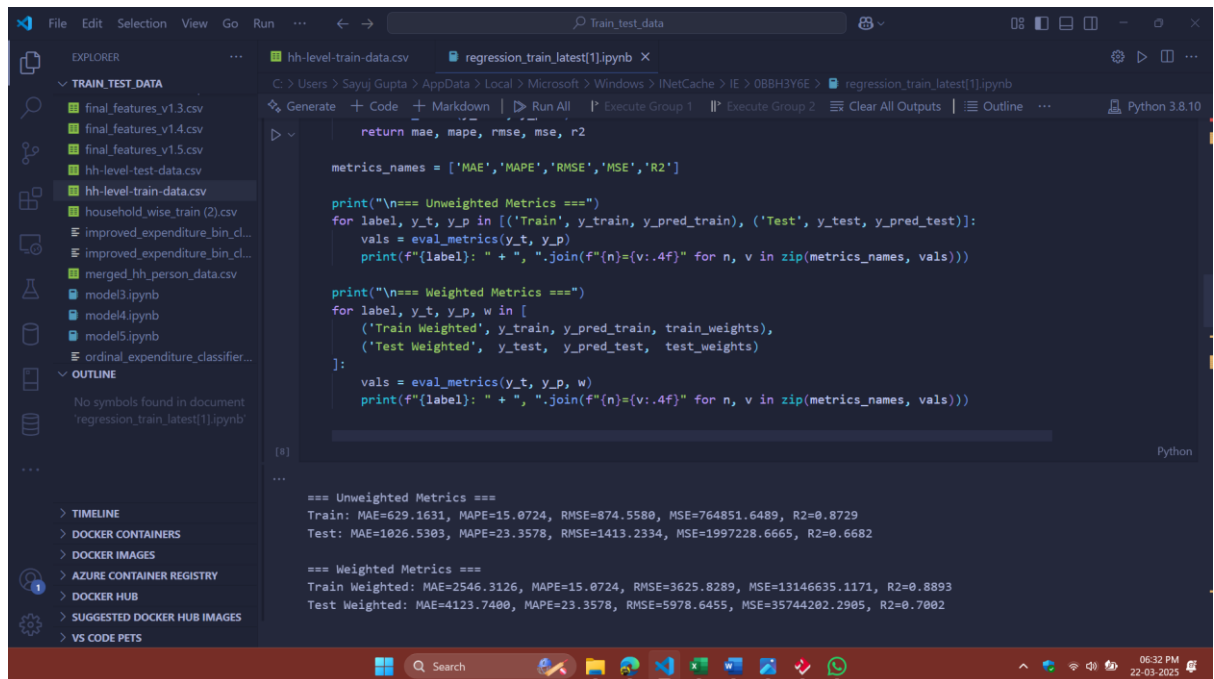
- For website:
 - i. index.html
 - ii. style.css
 - iii. script.js
 - iv. app.py
 - v. chat.py
 - vi. hh-level-train-data.csv
 - vii. hh-level-test-data.csv
 - viii. xgb_regressor_model_below95.pkl
 - ix. xgb_regressor_model_above95.pkl
 - x. rf_classification_model.pkl
- For Training:
 - i. hh-level-train-data.csv
 - ii. hh-level-test-data.csv
 - iii. person-level-train-data.csv
 - iv. person-level-test-data.csv
 - v. data_preprocessing.ipynb
 - vi. regression_train.ipynb
 - vii. classification_train.ipynb

G) Steps to execute:

- For Training:
 - i. Open the data_preprocessing.ipynb file.
 - ii. Ensure that the train and test datasets path are defined properly.
 - iii. Run the file to the end, thus creating pre-processed train and test datasets.
 - iv. Open the classification_train.ipynb file.
 - v. Use the pre-processed datasets to train the classifier which will classify between above and below 95%ile of Total Expenses.
 - vi. Open the regression_train.ipynb file.
 - vii. Use the preprocessed datasets to train both regressors.
- For website:
 - i. Ensure index.html, script.js, style.css, app.py and chat.py are in same directory.
 - ii. Run app.py
 - iii. Open index.html in web browser.

H) Screenshots of Execution:

Regression below 95%ile:



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'TRAIN_TEST_DATA' containing several CSV files and Jupyter notebooks. The code editor displays a Python script that calculates various metrics for training and testing data. The output of the script is shown in the bottom right corner.

```
return mae, mape, rmse, mse, r2

metrics_names = ['MAE', 'MAPE', 'RMSE', 'MSE', 'R2']

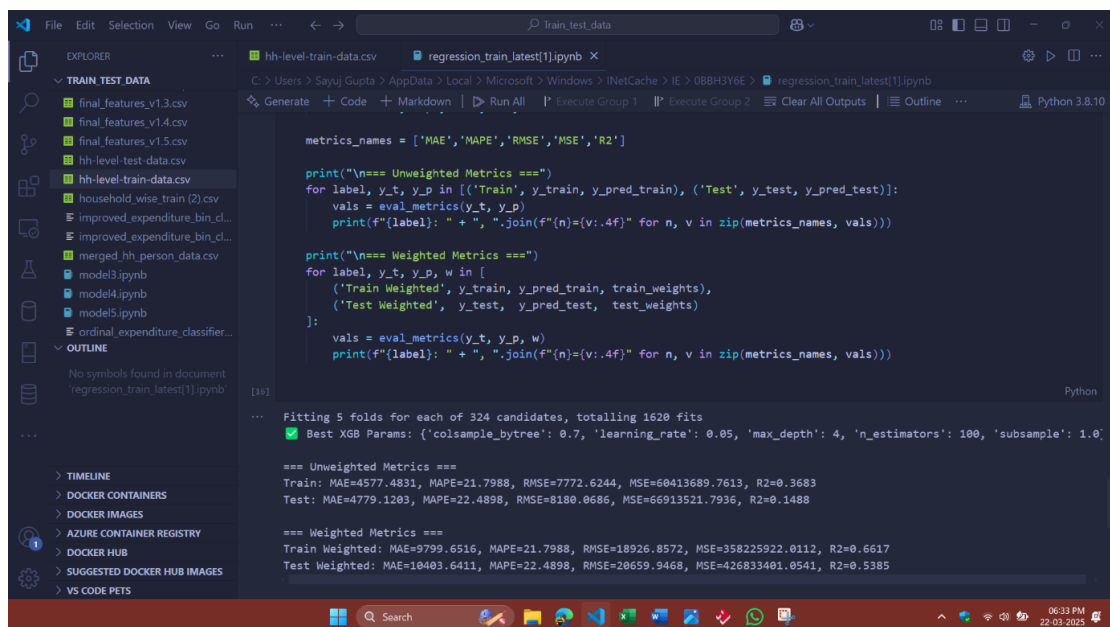
print("\n=== Unweighted Metrics ===")
for label, y_t, y_p in [('Train', y_train, y_pred_train), ('Test', y_test, y_pred_test)]:
    vals = eval_metrics(y_t, y_p)
    print(f"{label}: " + ", ".join(f"{n}={v:.4f}" for n, v in zip(metrics_names, vals)))

print("\n=== Weighted Metrics ===")
for label, y_t, y_p, w in [
    ('Train Weighted', y_train, y_pred_train, train_weights),
    ('Test Weighted', y_test, y_pred_test, test_weights)
]:
    vals = eval_metrics(y_t, y_p, w)
    print(f"{label}: " + ", ".join(f"{n}={v:.4f}" for n, v in zip(metrics_names, vals)))
```

=== Unweighted Metrics ===
Train: MAE=629.1631, MAPE=15.0724, RMSE=874.5580, MSE=764851.6489, R2=0.8729
Test: MAE=1026.5303, MAPE=23.3578, RMSE=1413.2334, MSE=1997228.6665, R2=0.6682

=== Weighted Metrics ===
Train Weighted: MAE=2546.3126, MAPE=15.0724, RMSE=3625.8289, MSE=13146635.1171, R2=0.8893
Test Weighted: MAE=4123.7400, MAPE=23.3578, RMSE=5978.6455, MSE=35744202.2905, R2=0.7002

Regression above 95%ile:



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'TRAIN_TEST_DATA' containing several CSV files and Jupyter notebooks. The code editor displays a Python script that calculates various metrics for training and testing data. The output of the script is shown in the bottom right corner.

```
metrics_names = ['MAE', 'MAPE', 'RMSE', 'MSE', 'R2']

print("\n=== Unweighted Metrics ===")
for label, y_t, y_p in [('Train', y_train, y_pred_train), ('Test', y_test, y_pred_test)]:
    vals = eval_metrics(y_t, y_p)
    print(f"{label}: " + ", ".join(f"{n}={v:.4f}" for n, v in zip(metrics_names, vals)))

print("\n=== Weighted Metrics ===")
for label, y_t, y_p, w in [
    ('Train Weighted', y_train, y_pred_train, train_weights),
    ('Test Weighted', y_test, y_pred_test, test_weights)
]:
    vals = eval_metrics(y_t, y_p, w)
    print(f"{label}: " + ", ".join(f"{n}={v:.4f}" for n, v in zip(metrics_names, vals)))
```

... Fitting 5 folds for each of 324 candidates, totalling 1620 fits
✔ Best XGB Params: {'colsample_bytree': 0.7, 'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 100, 'subsample': 1.0}

=== Unweighted Metrics ===
Train: MAE=4577.4831, MAPE=21.7988, RMSE=7772.6244, MSE=60413689.7613, R2=0.3683
Test: MAE=4779.1203, MAPE=22.4898, RMSE=8180.0686, MSE=66913521.7936, R2=0.1488

=== Weighted Metrics ===
Train Weighted: MAE=9799.6516, MAPE=21.7988, RMSE=18926.8572, MSE=358225922.0112, R2=0.6617
Test Weighted: MAE=10403.6411, MAPE=22.4898, RMSE=20659.9468, MSE=426833401.0541, R2=0.5385

Classification:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pickle

train_df = pd.read_csv("/home/puru/Downloads/final_train_v3.csv")
test_df = pd.read_csv("/home/puru/Downloads/final_test_v3.csv")

threshold = train_df['Total_Expense'].quantile(0.95)
train_df['is_top_5_percent'] = (train_df['Total_Expense'] >= threshold).astype(int)

X = train_df.drop(columns=['Total_Expense', 'is_top_5_percent'])
y = train_df['is_top_5_percent']

X = X.drop(columns=[col for col in X.columns if 'id' in col.lower()], errors='ignore')
test_X = test_df[X.columns]
test_df['is_top_5_percent'] = (test_df['Total_Expense'] >= threshold).astype(int)

model = RandomForestClassifier(
    n_estimators=100,
    max_depth=6,
    random_state=42
)

model.fit(X, y)

test_preds = model.predict(test_X)

y_test = test_df['is_top_5_percent']
accuracy = accuracy_score(y_test, test_preds)
print(f"✅ Accuracy on Test Set: {accuracy:.4f}")

with open('random_forest_model2.pkl', 'wb') as f:
    pickle.dump(model, f)
```

✓ 16.5s Python

✅ Accuracy on Test Set: 0.9596

Contact Details of Team Members:

1) Sayuj Gupta

- a. Email ID: 2023ucs0112@iitjammu.ac.in
- b. Mobile No.: 9464747414
- c. Institute: Indian Institute of Technology Jammu
- d. Course: Computer Science and Engineering
- e. Year of passing: 2027
- f. Contribution:
 - i. Team Leader
 - ii. Feature Engineering and Model Testing
 - iii. Presenter

2) HIMANSHU

- a. Email ID: 2023ucs0092@iitjammu.ac.in
- b. Mobile No.: 8595756912
- c. Institute: Indian Institute of Technology Jammu
- d. Course: Computer Science and Engineering
- e. Year of passing: 2027
- f. Contribution:
 - i. Website builder

ii. Feature Engineering

3) D Barghav

- a. Email ID: 2023ume0253@iitjammu.ac.in
- b. Mobile No.: 9790772220
- c. Institute: Indian Institute of Technology Jammu
- d. Course: Mechanical Engineering
- e. Year of passing: 2027
- f. Contribution:
 - i. Dataset preparation
 - ii. Model Training

4) Purushartha Gupta

- a. Email ID: 2023uce0062@iitjammu.ac.in
- b. Mobile No.: 9369573768
- c. Institute: Indian Institute of Technology Jammu
- d. Course: Civil Engineering
- e. Year of passing: 2027
- f. Contribution:
 - i. Dataset preparation
 - ii. Model Training

5) Parth Sachdeva

- a. Email ID: 2023ucs0105@iitjammu.ac.in
- b. Mobile No.: 9149663246
- c. Institute: Indian Institute of Technology Jammu
- d. Course: Computer Science and Engineering
- e. Year of passing: 2027
- f. Contribution:
 - i. Dataset preparation
 - ii. Model Training
 - iii. Feature Engineering