# Hackathon Report
# AI2000
# Foundations of Machine Learning

**Darpan Gaur**
**CO21BTECH11004**

**Bishwashri Roy**
**CS24RESCH11013**

## Preprocessing

For this dataset, we performed the following steps for handling missing values in the dataset:

- Checked for the presence of missing values in the dataset.
- We removed any columns that had more than 10% missing values, as they were not useful for the analysis.
- For the remaining missing values, we used the K-Nearest Neighbors (KNN) imputation method.
- KNN imputation fills in missing values by looking at the values of similar rows in the dataset.
- Estimates the missing values based on the values of k-nearest neighbors. [k=2]

Performed standard scaling to normalize the features in the dataset. Standard scaling involves transforming the data so that each feature has a mean of 0 and a standard deviation of 1.
Dropped 'UID', 'TownId', 'DistrictId' columns as they were not useful for the analysis.

### Miscellaneous

Tried below mentioned strategies, but not used in the final model as no significant improvement was observed.

- Used SMOTE oversampling to handle class imbalance, by generating synthetic samples for the minority classes ['low', 'high'].

- Used feature importance [Random Forest] to select the most important features for the models.
- Filling null values with mean, median, and mode.
- Used PCA to reduce the dimensionality of the dataset.
- Undersampling the majority class ['medium'] data.

## Models

For all models, used class weights - ['low': 3.0, 'medium': 1.0, 'high': 3.0] to handle class imbalance. Used grid search with cross-validation to tune hyperparameters.
We trained the following models on the preprocessed dataset:

### Random Forest

Hyperparameters tuned:

- n_estimators: Number of trees in the forest
- max_depth: Maximum depth of the tree
- min_samples_split: Minimum number of samples required to split an internal node
- min_samples_leaf: Minimum number of samples required to be at a leaf
- max_features: Number of features to consider when looking for the best split

### CatBoost

Hyperparameters tuned:

- iterations: Number of boosting iterations
- learning_rate: Boosting learning rate

### LightGBM

Hyperparameters tuned:

- n_estimators: Number of boosting rounds
- subsample: Subsample ratio of the training instance
- reg_lambda: L2 regularization term on weights
- reg_alpha: L1 regularization term on weights

- learning_rate: Boosting learning rate
- num_leaves: Maximum tree leaves for base learners
- max_depth: Maximum tree depth for base learners
- min_child_samples: Minimum number of data needed in a child
- colsample_bytree: Subsample ratio of columns when constructing each tree

### Ensemble

- Formed an ensemble of the above models.
- Used soft voting to combine the predictions of the models.
- Used weights based on the performance (f1 score) of the individual models, obtained by cross-validation on the training data.

### Miscellaneous

Also tried other models like Naive Bayes, AdaBoost, and XGBoost, but they did not perform as well as the models mentioned above.
Used neural networks (MLP) for regression, but the performance was not satisfactory.

## Results

F1 score used as the evaluation metric for the models.
Found by cross-validation with 10 folds on the training data.

| **Model** | **F1 Score** |
|---|---|
| Random Forest | 0.4274 |
| CatBoost | 0.4280 |
| LightGBM | 0.4281 |
| Ensemble | 0.433 |

Table 1: F1 Score of the models

## Individual Contributions (Darpan Gaur)

Trained and tuned CatBoost, LightGBM, XGBoost and AdaBoost models.
Performed feature importance analysis and feature selection.
Used PCA to reduce the dimensionality of the dataset.
Performed oversampling using SMOTE to handle class imbalance.
Performed ndersampling to handle class imbalance.