Name:

Yashank Rajvanshi

Project Title:

Automated Character Recognition with Anomaly Detection

Business/Problem Context:

Optical Character Recognition (OCR) is widely used in document digitization, postal sorting, license plate recognition, and similar applications. The challenge is that the same character can appear differently due to variations in fonts, distortions, or scanning artifacts. Misclassification can lead to costly errors (e.g., misrouted mail, incorrect data entry). Additionally, unusual or distorted character shapes (anomalies) may not be recognized well, reducing the reliability of the system. Hence, there is a business need for a robust OCR system that not only classifies letters correctly but also flags distorted or anomalous shapes for review.

Objective Statement:

The objective of this project is to **build a machine learning model that can classify black-and-white character images into one of 26 English capital letters with high accuracy (≥97%)** while also **detecting anomalous or distorted shapes that deviate from normal patterns**. This dual approach will improve both the **accuracy** and the **reliability** of OCR systems in real-world deployments.

Problem Type:

This is a **Supervised Classification Problem**.

- **Goal:** Predict which English alphabet letter (A–Z) a given character image belongs to.
- **Secondary Task:** Detect anomalous character samples that deviate from typical letter patterns.

Target Variable & Key Inputs:

- **Target Variable (y):** The **letter label** (categorical, 26 classes: A–Z).
- **Key Inputs (X):** Numeric features extracted from character images (e.g., stroke counts, shape descriptors, pixel statistics).
    - Example features: curvature, edges, number of horizontal/vertical lines, symmetry, pixel density, etc.
    - Dataset is structured with each row = one letter sample, columns = extracted features.

Assumptions:

- The dataset is representative of real-world handwriting/printed character variations.
- Anomalies are relatively rare compared to normal samples.
- Features are already engineered from images (no raw image preprocessing in this scope).
- Scaling/encoding has been handled consistently before model training & deployment.

Dataset Source:

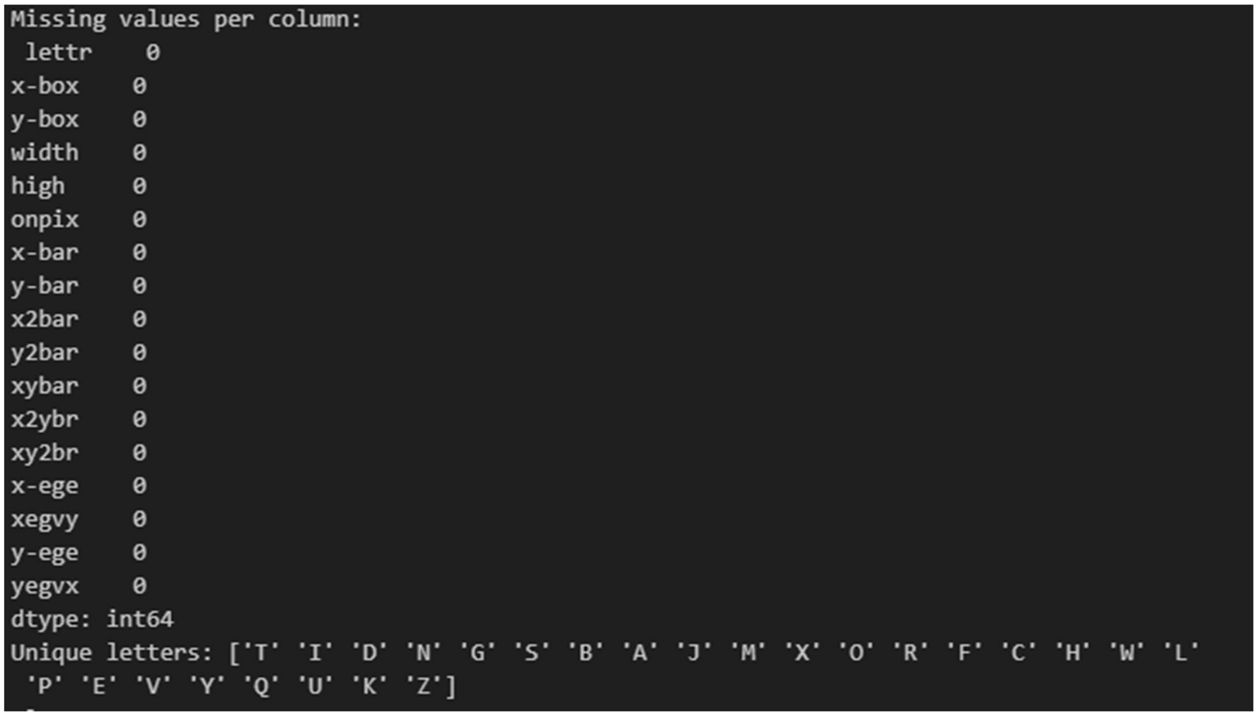
Link - letter-recognition-data.xlsx (As given by administrator)

- Dataset was quite clean & accurate
- But the size was too less to reach a more accurate result.
- Also the outliers detected, included the data, which helped improve our model only.

Data Description:

- Shape - (20000, 17)
- Features - [‘lettr’, ‘x-box’, ‘y-box’, ‘width’, ‘high’, ‘onpix’, ‘x-bar’, ‘y-bar’, ‘x2bar’, ‘y2bar’, ‘xybar’, ‘x2ybar’, ‘xy2bar’, ‘x-ege’, ‘xegvy’, ‘y-ege’, ‘yegvx’]

- Missing Value Check –

- 

```
Missing values per column:
 lettr    0
x-box    0
y-box    0
width    0
high     0
onpix    0
x-bar    0
y-bar    0
x2bar    0
y2bar    0
xybar    0
x2ybr    0
xy2br    0
x-ege    0
xegvy    0
y-ege    0
yegvx    0
dtype: int64
Unique letters: ['T' 'I' 'D' 'N' 'G' 'S' 'B' 'A' 'J' 'M' 'X' 'O' 'R' 'F' 'C' 'H' 'W' 'L'
 'P' 'E' 'V' 'Y' 'Q' 'U' 'K' 'Z']
```

Encode Target

- The target variable in this dataset is the letter label (A–Z).
- Since machine learning models work with numbers (not text), we must convert categorical labels into numeric form.
- Example:
  - A → 0, B → 1, ..., Z → 25.
- This process is called Label Encoding, and it ensures the model interprets each class correctly.

Split Features & Target

- The dataset has 16 numerical features describing character shapes and 1 target column (letter).
- We separate:
    - X (Features): All numerical predictors like x-box, y-box, width, etc.
    - y (Target): The encoded letter label.
- This ensures a clean separation between input variables (used to make predictions) and the outcome we want to predict.

Scaling Features

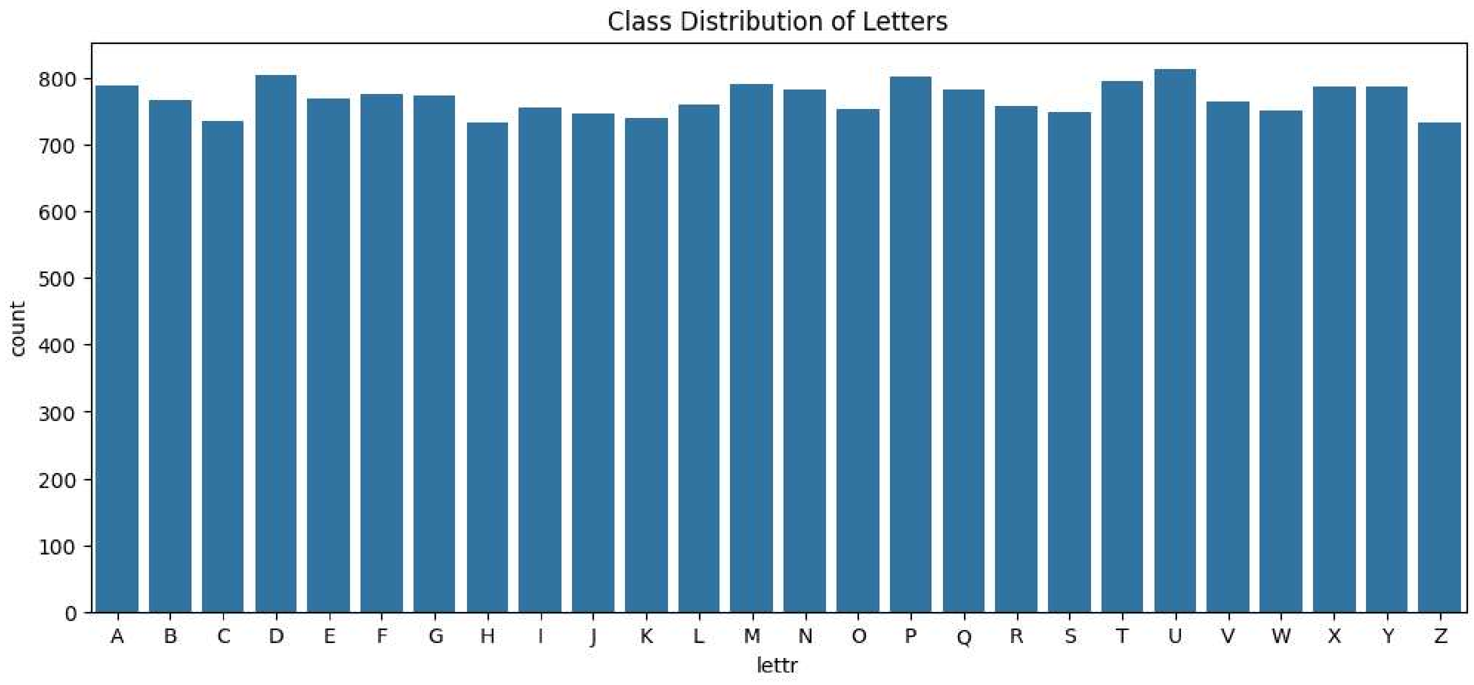
- Features like width, height, onpix, etc. have different ranges.
    - Example: width might range from 1–15, while onpix might range from 0–100.
- Models like Logistic Regression, KNN, and Neural Networks are sensitive to feature scale.
- Therefore, we apply Standardization (z-score scaling) or Normalization to bring all features onto a similar scale.
- This improves training stability and ensures no single feature dominates the model just because of its scale.

Train-Test Split

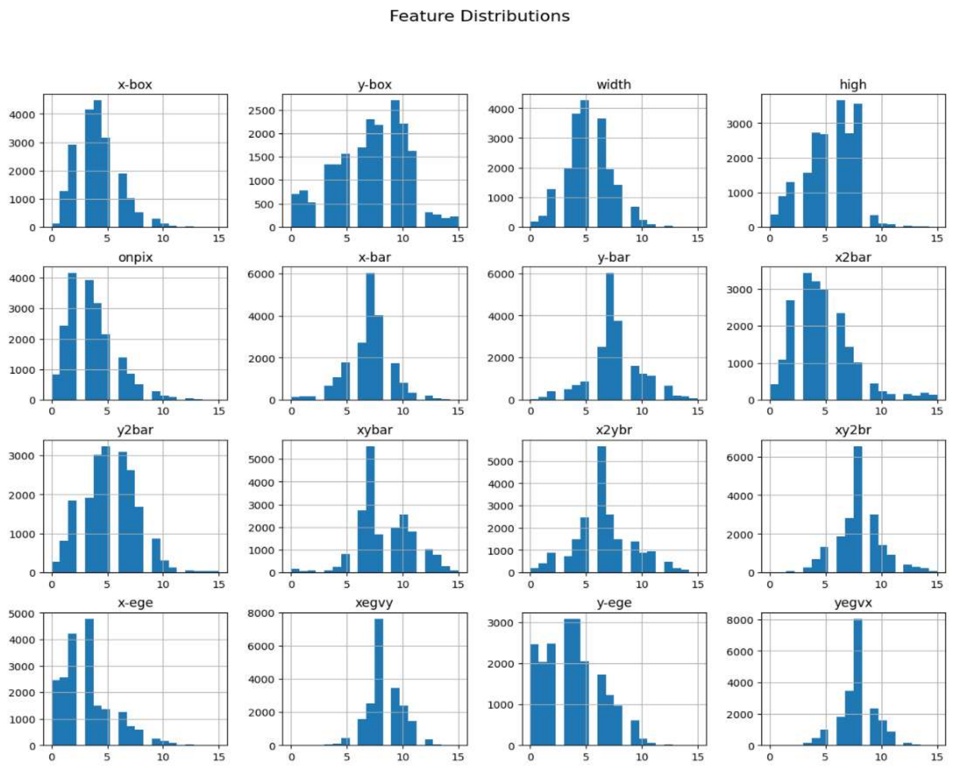
- To evaluate the model fairly, we split the dataset into:
    - Training set (e.g., 80%) → used to fit the model.
    - Test set (e.g., 20%) → used to evaluate model performance on unseen data.
- This prevents overfitting and gives a realistic estimate of how the model will perform in real-world usage.

Exploratory Data Analysis (EDA):

- Class Distribution:
  - Ensures the dataset is **balanced** (similar number of samples per letter).
  - If some letters are underrepresented, the model might learn poorly for them.



- **Feature Distributions:**
  - **Helps understand the range and spread of features.**
  - **Detects outliers (e.g., unusually high width values)**



PCA Based Anomaly Detection:

1. Standardize Features

- Since PCA is scale-sensitive, we first scale all features (e.g., z-score normalization).

2. Apply PCA

- Fit PCA on the dataset to extract principal components.
- Keep only the top k components (e.g., those that explain 95% of variance).

3. Reconstruct Data

- Transform original data into principal component space.
- Then reconstruct back to the original feature space using only top k components.

4. Calculate Reconstruction Error

- Normal samples will have low error (since PCA captured their structure).
- Anomalous samples will have high error (since they don’t align with main patterns).

5. Set Threshold

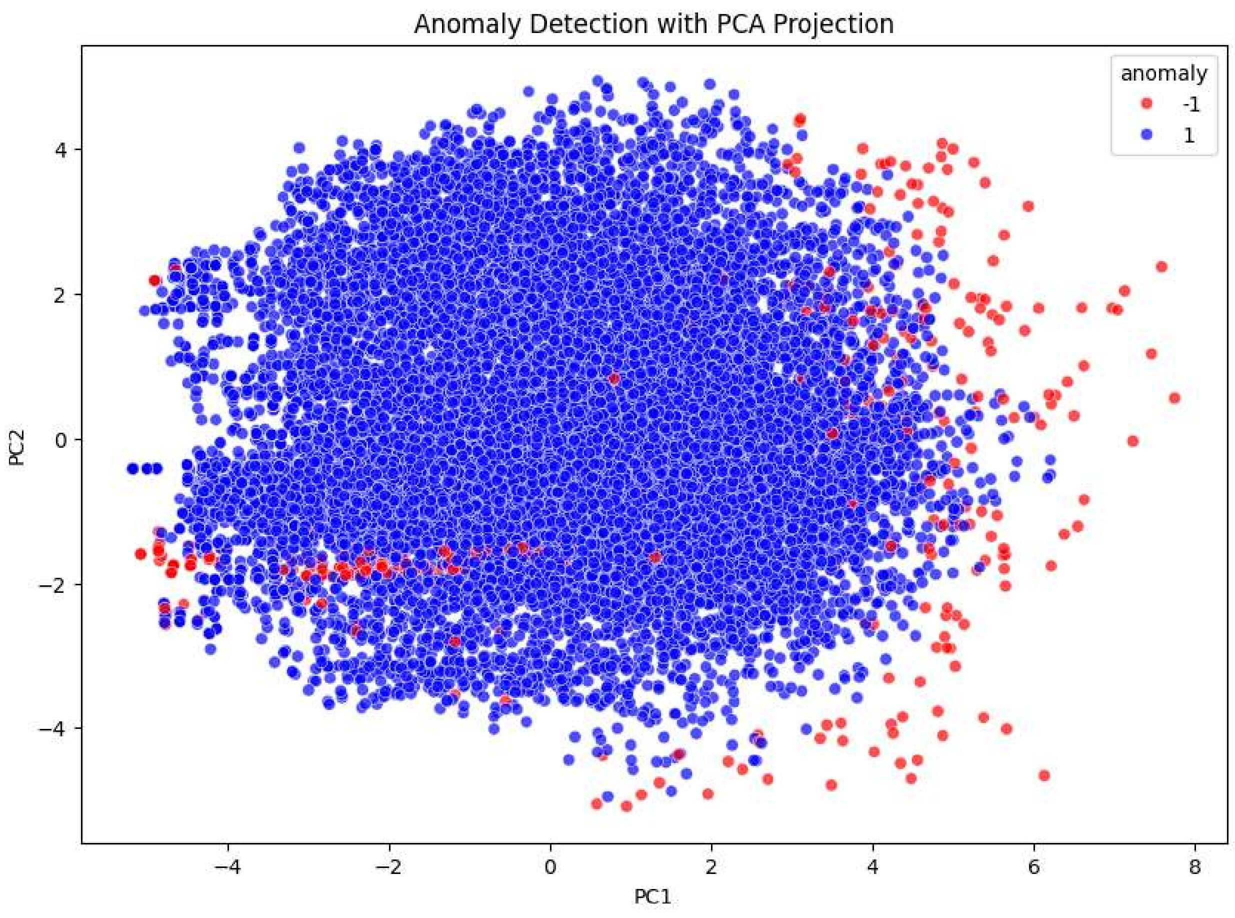
- Decide a cutoff (e.g., mean + 2 standard deviations, or top 5% error values).
- Samples with error above threshold are flagged as anomalies.

Why PCA Only?

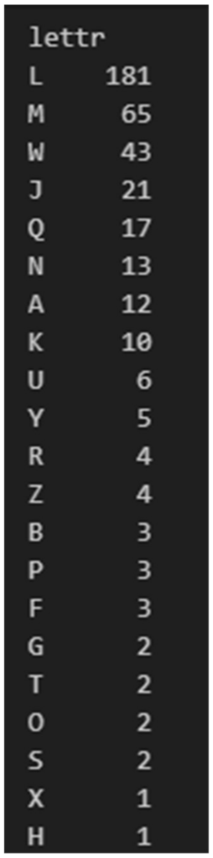
- An unsupervised method chosen because anomalies are not labeled.
- Works by detecting samples that deviate from the main variance structure of the dataset.
- Lightweight, interpretable, and effective for **dimensionality reduction + anomaly scoring**.

⇨ For the **letter recognition dataset**:

- Normal letters (A–Z in common fonts and distortions) will align with the main PCA structure.
- Rare or unusually distorted letters will have **higher reconstruction error**, so PCA can catch them as anomalies.



Anomalies Character-wise:



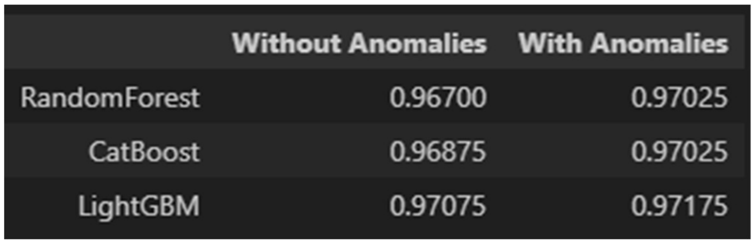
```
lettr
L    181
M     65
W     43
J     21
Q     17
N     13
A     12
K     10
U      6
Y      5
R      4
Z      4
B      3
P      3
F      3
G      2
T      2
O      2
S      2
X      1
H      1
```

⇨ **We are treating anomalies as Noise for now.**

Models:

1. **Random Forest Classifier (Baseline Model)**
   - Chosen because it is a **robust ensemble model** that works well on tabular data with mixed feature types.
   - It reduces overfitting through bagging and randomness.
   - Handles **nonlinear relationships** and **feature interactions** automatically.
   - Easy to interpret feature importance.
2. **CatBoost Classifier**
   - Selected as an **advanced gradient boosting algorithm** optimized for categorical and tabular data.
   - Provides **built-in regularization** and **fast convergence**.
   - Handles feature interactions better than Random Forest.
   - Known for requiring less hyperparameter tuning compared to other boosting methods.
3. **LightGBM Classifier**
   - Another gradient boosting framework, but highly efficient with **large datasets**.
   - Uses **leaf-wise tree growth**, which makes it faster and more accurate in many cases.
   - Well-suited for high-dimensional features like in this dataset.
   - Known for excellent performance in competitions and production OCR-like tasks.

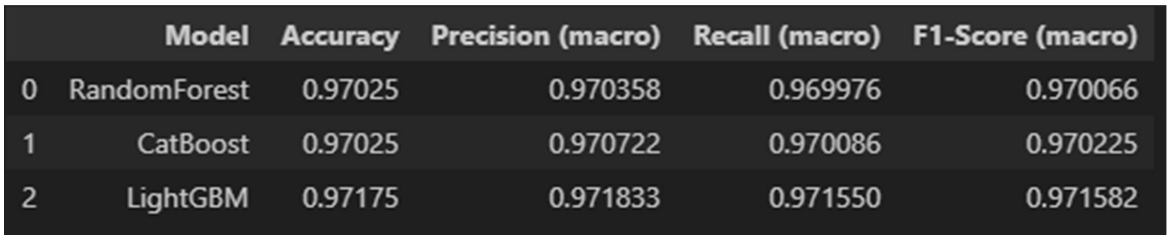
Evaluation Metrics:



	Without Anomalies	With Anomalies
RandomForest	0.96700	0.97025
CatBoost	0.96875	0.97025
LightGBM	0.97075	0.97175

Now, we can see that models with anomalies are performing better than the one's without anomalies. This means, some anomalies might have removed the positive results too, due to which the accuracy might have taken a hit.

Evaluation Metrics (With Anomalies):



	Model	Accuracy	Precision (macro)	Recall (macro)	F1-Score (macro)
0	RandomForest	0.97025	0.970358	0.969976	0.970066
1	CatBoost	0.97025	0.970722	0.970086	0.970225
2	LightGBM	0.97175	0.971833	0.971550	0.971582

Error Analysis:

- Misclassifications mainly occur between **visually similar letters**, such as:
  - O vs Q → both round, small tails can confuse the model.
  - I vs L → narrow vertical strokes, difficult to separate with limited features.
  - C vs G → open circular structures with subtle differences.
- Rare **distorted samples (anomalies)** increase error rates because the model was not trained to expect extreme variations.
- Errors may also arise from **feature overlap**, where numerical features (like x-box, y-box, width) are similar for multiple letters.