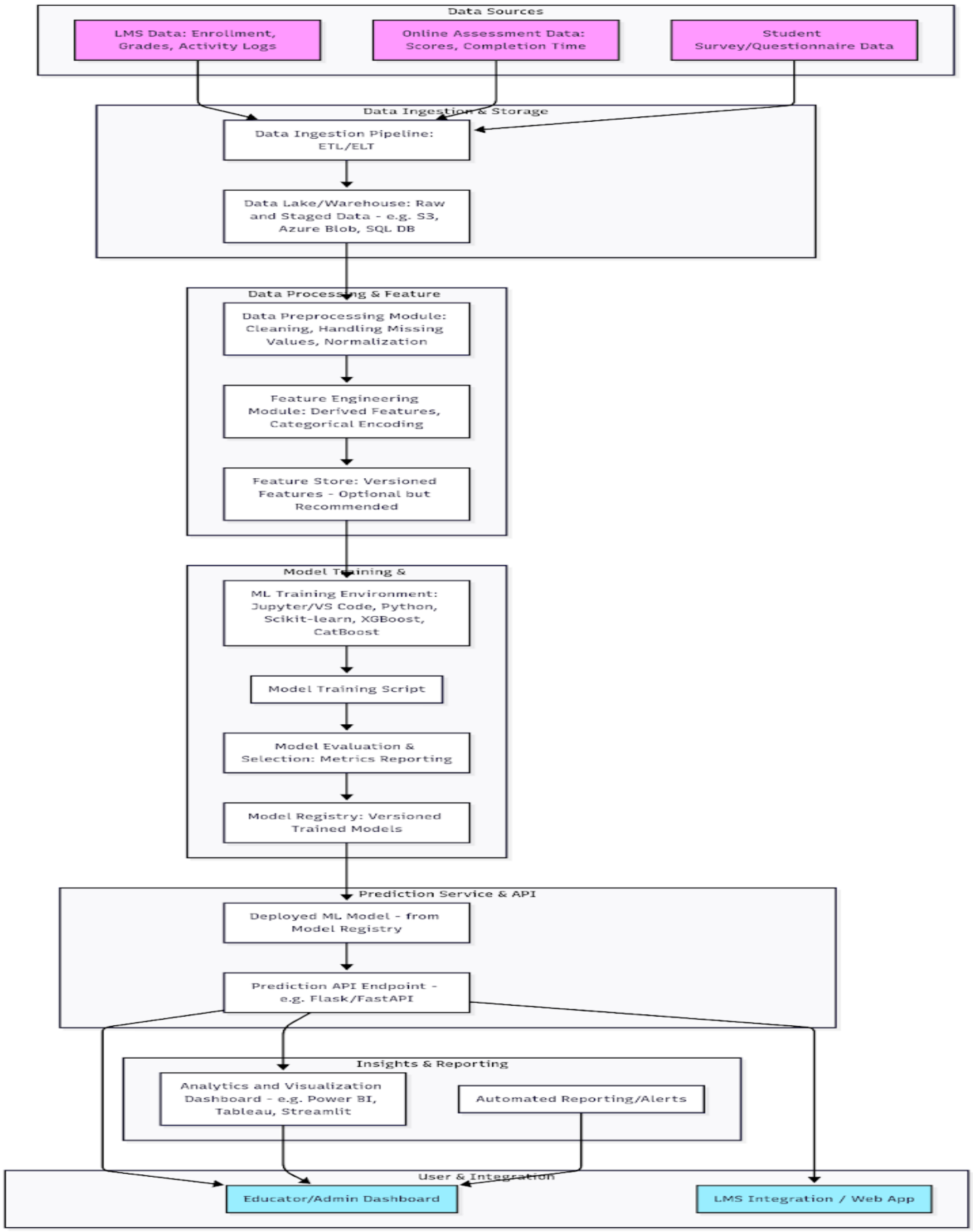
E-Adapt: Predicting Student Adaptability In Online Classes

The rapid advancement of technology has fundamentally revolutionized the landscape of education, ushering in an era where learning is no longer confined to traditional brick-and-mortar classrooms. This technological surge has led to the widespread adoption and sophisticated evolution of online learning platforms and e-learning tools. While this digital transformation undeniably offers immense benefits, such as unparalleled **flexibility** in scheduling and learning pace, and significantly enhanced **accessibility** for students across diverse geographical locations and personal circumstances, it simultaneously introduces a unique set of challenges that can profoundly impact student success.

Online classes demand a distinct set of skills and a different learning approach compared to conventional face-to-face instruction. Students are often thrust into an environment that necessitates a high degree of **self-motivation** to engage with course material without constant external prompts or in-person interactions. Effective **time management** becomes paramount, as learners must independently structure their study schedules, meet deadlines, and balance academic commitments with other responsibilities, often without the structured routine of a physical classroom. Furthermore, a foundational level of **technological proficiency** is indispensable; students must comfortably navigate virtual learning environments, utilize various digital tools for communication and assignments, troubleshoot technical issues, and effectively leverage online resources.

Crucially, not all students possess these inherent capabilities or can easily adapt to this self-directed, technologically-dependent digital environment. Many struggle with the transition, feeling isolated, overwhelmed by the lack of immediate support, or falling behind due to underdeveloped self-regulation skills. Without timely identification of these struggles and the provision of targeted support mechanisms, students are at a higher risk of disengagement, poor academic performance, or even dropping out, underscoring the critical need to understand and predict student adaptability in online learning.

Technical Architecture



Project Flow

- **User Interface Interaction**: We were able to successfully allow users to interact with the created User Interface (UI) for input entry.

- **Integrated Model Analysis**: The input entered was analyzed seamlessly by the integrated machine learning model.
- **Prediction Display**: After processing, the prediction made by the model was effectively displayed again on the UI, with instant results to the user.

Activities Completed

1. Problem Definition & Understanding:

- The particular business problem had been well defined.

2. Data Collection & Preparation:

- The required dataset for the project was successfully gathered.
- Thorough data preparation activities were carried out in order to provide quality data and make it ready for modeling.

3. Exploratory Data Analysis (EDA):

- Descriptive statistical analysis was conducted on the dataset.
- Visual examinations were done to reveal patterns, associations, and insights within the data.

4. Model Building:

- Multiple machine learning algorithms were used to train the predictive model.
- Exhaustive model testing was done to validate its initial performance.

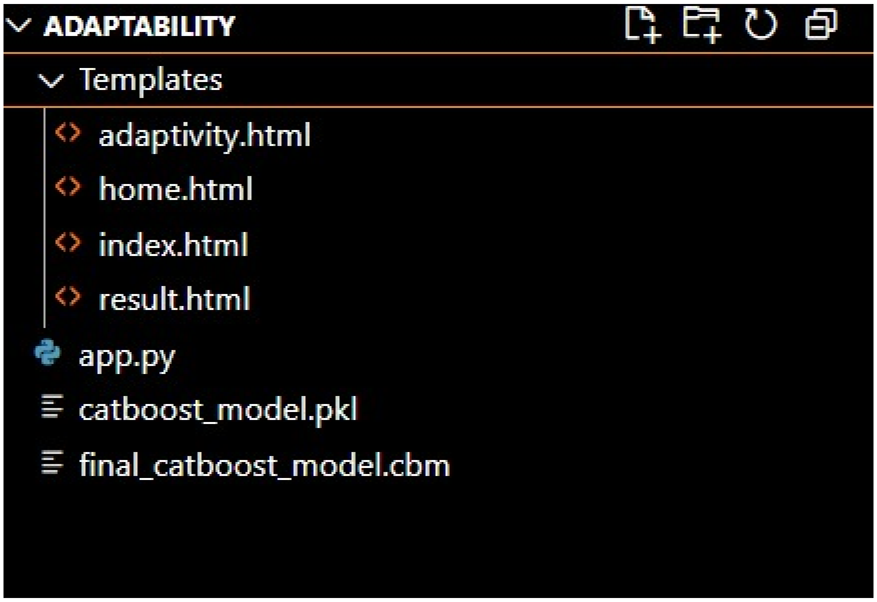
5. Performance Testing & Hyperparameter Tuning:

- The performance of the model was systematically tested under different evaluation criteria.
- Model accuracy was contrasted before and after hyperparameter tuning, with improvements shown.

6. Model Deployment:

- The top-performing model was saved effectively for deployment.
- The model was integrated with a web framework to develop a working web application.

Project Structure:



- We are building a flask application which needs HTML pages stored in the app.py.
- final_catboost_model,cbm is our saved model. Further we will use this model for flask integration.
- The index.html contains all the features. The result.html contains the results.
- The adaptivity.html contains predictions of the model.

Milestone 1: Define Problem / Problem Understanding

Activity 1: The Business Problem

With the rapid shift towards online learning, especially after the pandemic, educational institutions face challenges in understanding how well students adapt to virtual learning environments. Many students struggle with issues like poor connectivity, lack of self-regulation, and socio-economic barriers that directly affect their ability to adapt.

The business problem addressed by the **E-Adapt** project is to develop a machine learning model that can **predict the adaptability level (Low, Medium, High) of students** based on various demographic, socio-economic, and behavioral features. This will help institutions proactively identify students who need extra support, personalize interventions, and improve overall learning outcomes in online education.

Activity 2: Business Requirements

The E-Adapt project has the following key business requirements:

- **Accurate and Timely Predictions:** The model should use relevant and up-to-date data to ensure adaptability levels are predicted reliably and reflect current learning conditions.
- **Scalability and Flexibility:** The prediction system should be scalable to accommodate data from different institutions and flexible enough to adapt to new patterns and student behaviors as more data becomes available.
- **Privacy and Compliance:** The system must comply with data privacy policies and regulations regarding students' personal and academic information.
- **Actionable Insights:** The output should be easy to interpret by educators and decision-makers, enabling them to design targeted interventions, support plans, or policy changes.
- **User-friendly Interface:** The final system should have a clear and intuitive interface for stakeholders to view predictions, reports, and suggested actions.

Activity 3: Literature Survey

To learn more about the main elements that affect students' flexibility in online learning settings, the state of the field's research, and how predictive models might be developed to overcome this difficulty, a review of the literature was done.

Globally, the shift to online learning was greatly accelerated by the COVID-19 pandemic, underscoring the necessity of strong virtual education systems. The ability of students to adjust to this abrupt change has been the subject of numerous studies. According to research, a variety of demographic factors (such as age, gender, and location), socioeconomic status, digital literacy, motivation, self-discipline, and psychological preparedness all affect how adaptable an online learning environment is.

Important conclusions from the body of existing literature:

- Demographic and Socioeconomic Factors: Research like that of Pal and Vanijja (2020) highlights that family income, the availability of personal devices, and the type of internet are important factors that determine the success of online learning. Due to limited resources and poor connectivity, students from low-income or rural backgrounds have a harder time adjusting.

- Technology and Infrastructure: Rasheed et al. (2020) found that reliable internet access and appropriate devices (such as laptops, smartphones, and tablets) are necessary for successful online learning participation. Low bandwidth connections and frequent power outages (also known as load shedding) are significant obstacles, especially in developing nations.
- Psychological and Behavioural Readiness: It is frequently noted that self-directed learning aptitude, drive, and ease of use of Learning Management Systems (LMS) are powerful indicators of adaptability.
- Current Predictive Models: Prior research has investigated the use of machine learning to forecast online learning dropout rates and student performance. In educational data mining, algorithms such as Random Forests, Decision Trees, Neural Networks, and Logistic Regression have been extensively employed. Fewer studies, nevertheless, have explicitly examined predicting students' levels of adaptability, which integrates resilience and engagement factors.

Gaps Found:

- Performance prediction is the subject of many studies, but adaptability is not one of them.
- There is little research on combining behavioural data with socioeconomic and infrastructure factors.
- Institutions' inability to easily understand and respond to these forecasts in real time.

Relevance to our E-Adapt project: To fill these gaps, the E-Adapt project creates a machine learning model that predicts adaptability levels (Low, Medium, High) by combining behavioural, technological, and demographic characteristics. The model seeks to improve inclusivity and educational outcomes by giving educators practical insights to help students who are having difficulty with online learning.

Milestone 2: Data Collection & Preparation

Activity 1: Collect the dataset

Dataset link: https://www.kaggle.com/datasets/mdmahmudulhasansuzan/students-adaptability-level-in-online-education

Activity 1.1: Importing all the libraries:

```
import numpy as np import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, Gradient Boosti from sklearn.tree import Decision TreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
```

Activity 1.2: Read the Dataset

	Gender	Age	Education Level	Institution Type	IT Student	Location	Load-shedding	Financial Condition	Internet Type	Network Type	Class Duration	Self Lms	Device	Adaptivity Level
0	Boy	21-25	University	Non Government	No	Yes	Low	Mid	Wifi	4G	3-6	No	Tab	Moderate
1	Girl	21-25	University	Non Government	No	Yes	High	Mid	Mobile Data	4G	1-3	Yes	Mobile	Moderate
2	Girl	16-20	College	Government	No	Yes	Low	Mid	Wifi	4G	1-3	No	Mobile	Moderate
3	Girl	11-15	School	Non Government	No	Yes	Low	Mid	Mobile Data	4G	1-3	No	Mobile	Moderate
4	Girl	16-20	School	Non Government	No	Yes	Low	Poor	Mobile Data	3G	0	No	Mobile	Low

Activity 2: Data Preparation

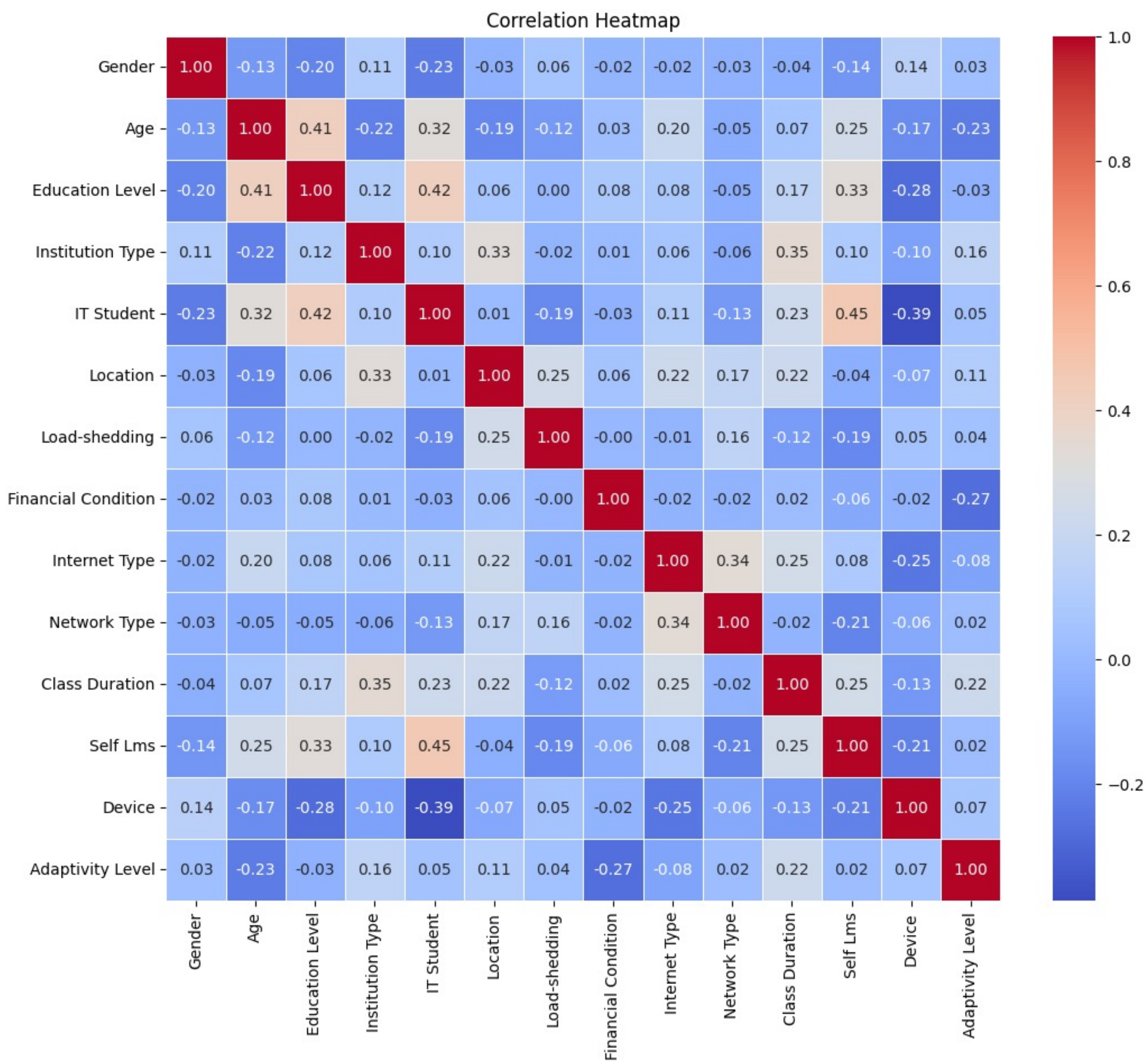
- Handling missing values
- Handling Outliers

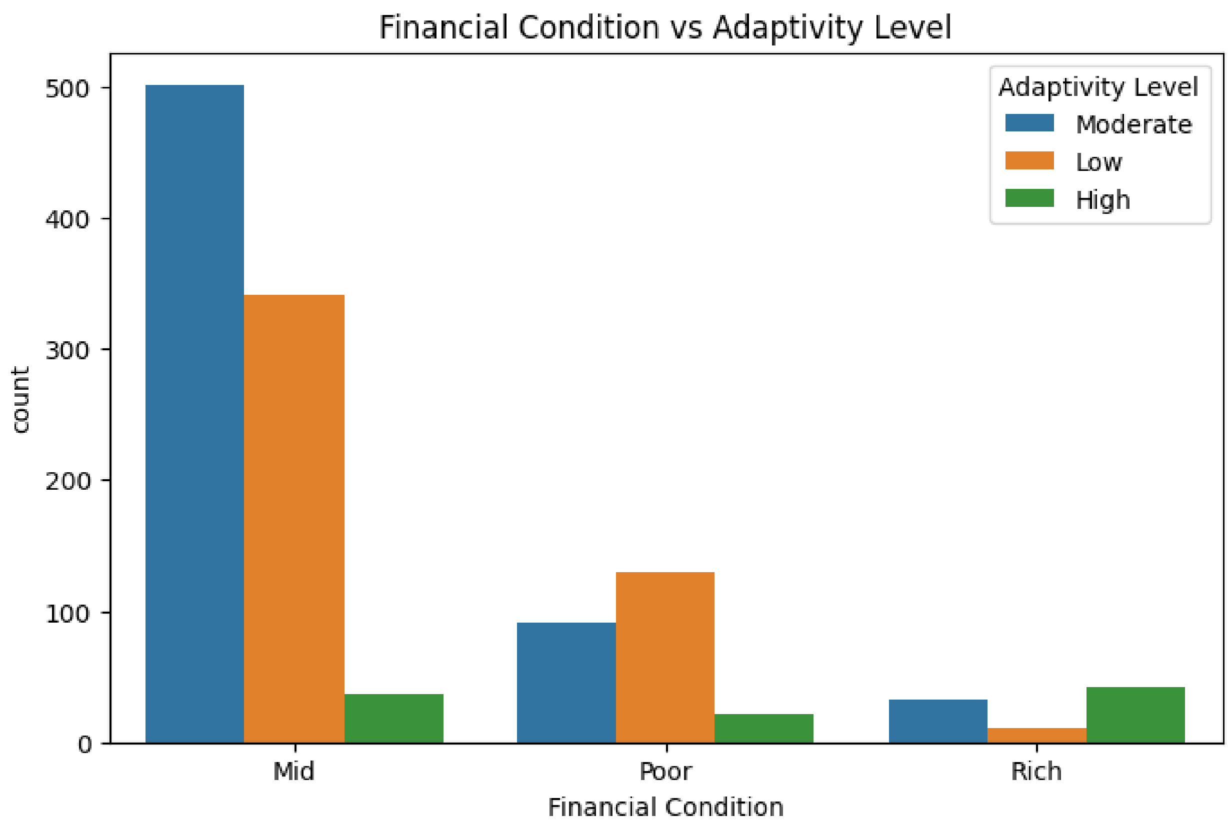
Activity 2.1: Handling missing values

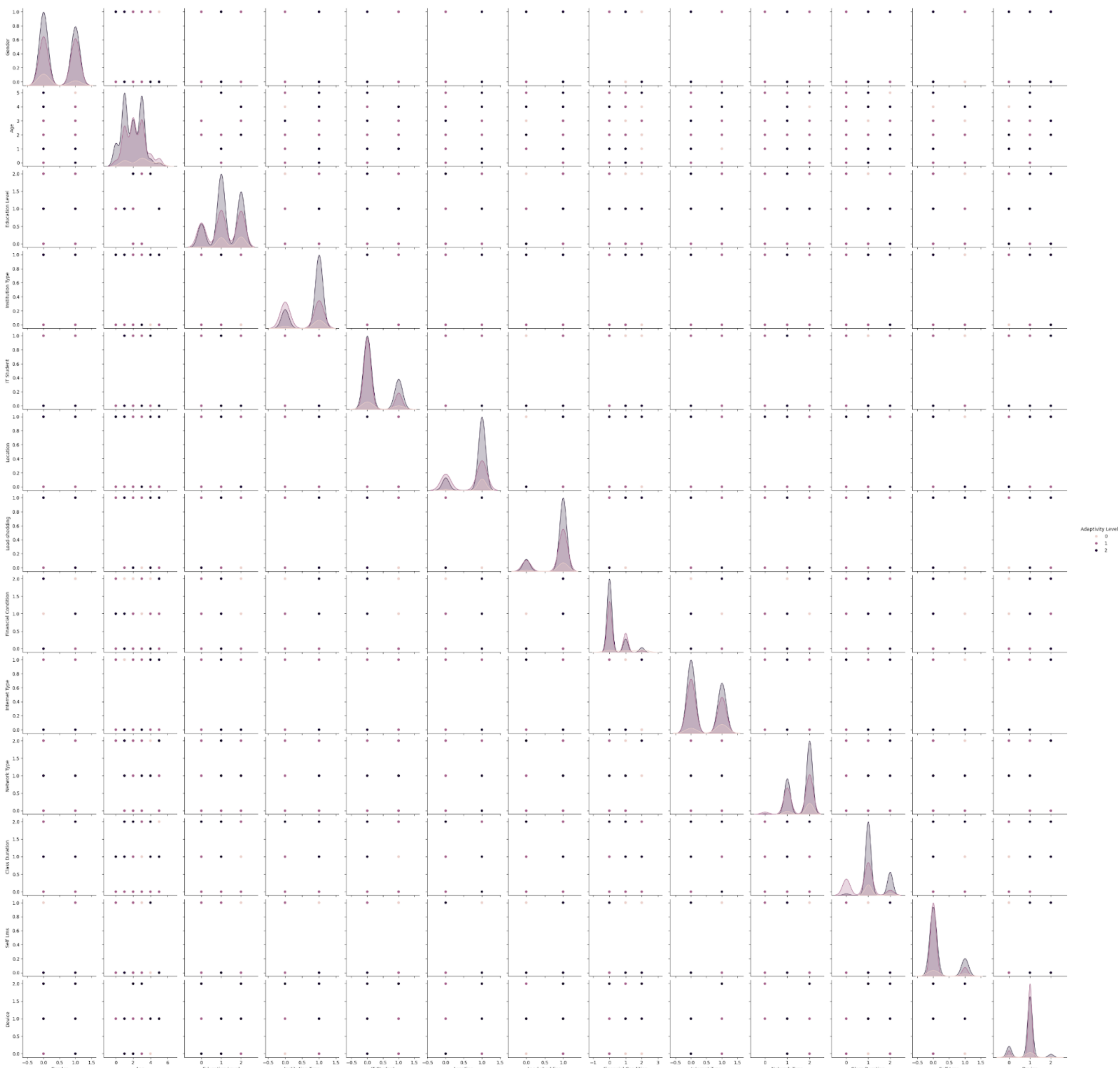
```
RangeIndex: 1205 entries, 0 to 1204
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Gender               1205 non-null   object
 1   Age                  1205 non-null   object
 2   Education Level      1205 non-null   object
 3   Institution Type     1205 non-null   object
 4   IT Student           1205 non-null   object
 5   Location             1205 non-null   object
 6   Load-shedding        1205 non-null   object
 7   Financial Condition  1205 non-null   object
 8   Internet Type        1205 non-null   object
 9   Network Type         1205 non-null   object
 10  Class Duration       1205 non-null   object
 11  Self Lms             1205 non-null   object
 12  Device               1205 non-null   object
 13  Adaptivity Level     1205 non-null   object
dtypes: object(14)
memory usage: 131.9+ KB
None
Gender                 0
Age                    0
Education Level        0
Institution Type       0
IT Student             0
Location               0
Load-shedding          0
Financial Condition    0
Internet Type          0
Network Type           0
Class Duration         0
Self Lms               0
Device                 0
Adaptivity Level       0
dtype: int64
```

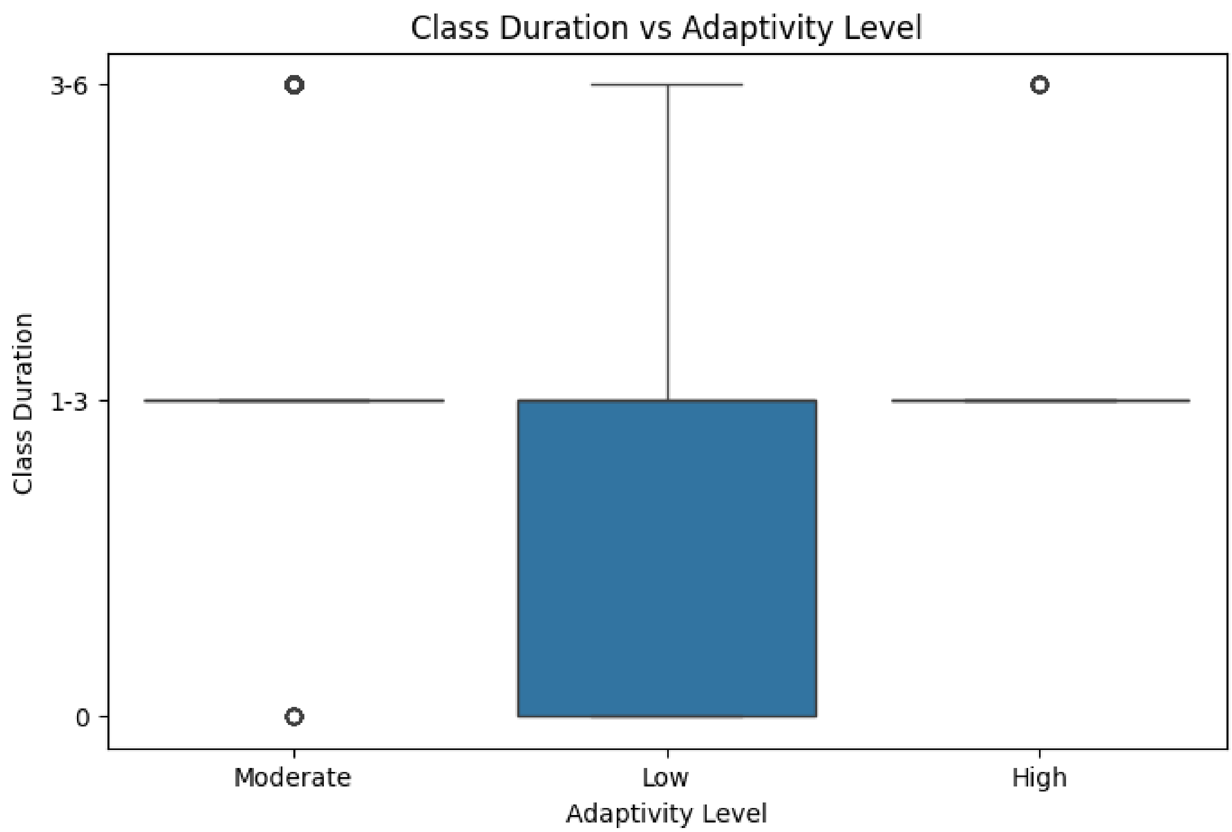
Milestone 3: Exploratory Data Analysis

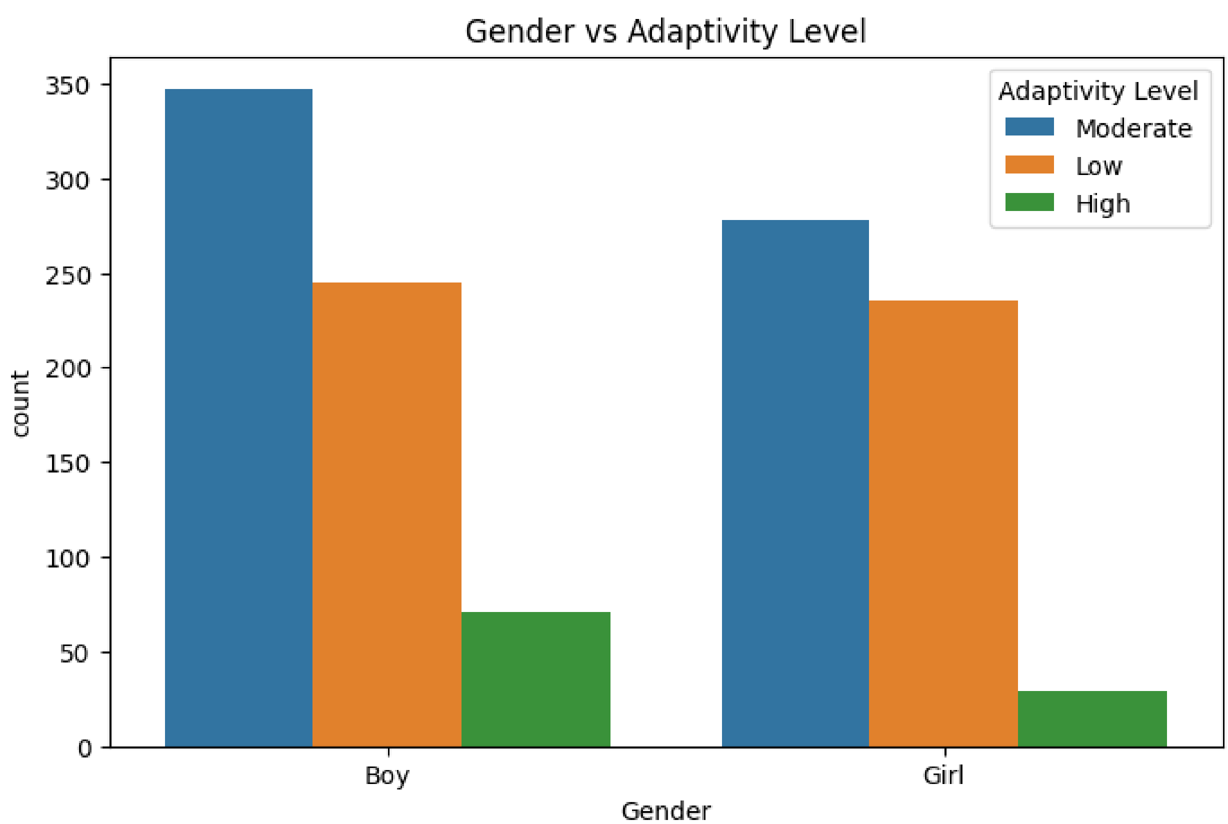
Activity 1: Descriptive statistical

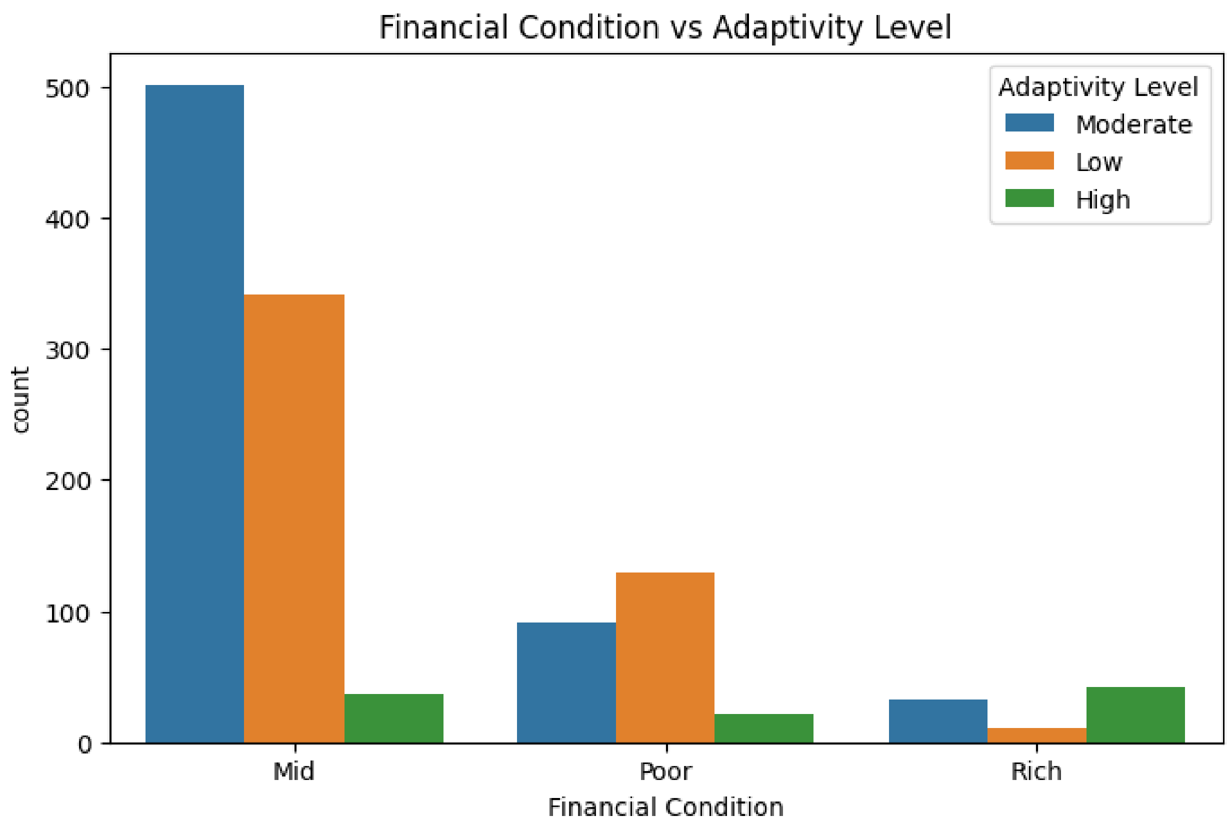

Correlation Heatmap
Gender
Age
Education Level
Institution Type
IT Student
Location
Load-shedding
Financial Condition
Internet Type
Network Type
Class Duration
Self Lms
Device
Adaptivity Level


Financial Condition vs Adaptivity Level
Adaptivity Level
Moderate
Low
High
count
500
400
300
200
100
0
Mid
Poor
Rich
Financial Condition


Gender
Age
Education Level
Institution Type
IT Student
Location
Load-shedding
Financial Condition
Internet Type
Network Type
Class Duration
Self Lms
Device
Adaptivity Level
0
1
2


Class Duration vs Adaptivity Level
3-6
1-3
0
Class Duration
Moderate
Low
High
Adaptivity Level


Gender vs Adaptivity Level
350
300
250
200
150
100
50
0
count
Adaptivity Level
Moderate
Low
High
Boy
Girl
Gender



Encoding the Categorical Features:

```python
def classifiers_modeling(classifiers, X, y, features):
    results = []

    preprocessor = ColumnTransformer(
        transformers=[('cat', OneHotEncoder(handle_unknown='ignore'), features)]
    )


    kf = KFold(n_splits=CONFIG.FOLD, shuffle=True, random_state=CONFIG.RANDOM_STATE)

    for name, model in classifiers.items():
        scores = []
        for train_idx, valid_idx in kf.split(X):
            X_train, X_valid = X.iloc[train_idx], X.iloc[valid_idx]
            y_train, y_valid = y[train_idx], y[valid_idx]


            X_train_enc = preprocessor.fit_transform(X_train)
            X_valid_enc = preprocessor.transform(X_valid)

            X_train_enc, X_valid_enc = _scale(X_train_enc, X_valid_enc,
                                              list(range(X_train_enc.shape[1])))

            model.fit(X_train_enc, y_train)
            preds = model.predict(X_valid_enc)
            acc = accuracy_score(y_valid, preds)
            scores.append(acc)

        avg_score = np.mean(scores)
        print(f"{name}: {avg_score:.4f}")
        results.append({'Model': name, 'Accuracy': avg_score})
```

Splitting data into train and test

```
def classifiers_modeling(classifiers, X, y, features):
    results = []

    preprocessor = ColumnTransformer(
        transformers=[('cat', OneHotEncoder(handle_unknown='ignore'), features)]
    )


    kf = KFold(n_splits=CONFIG.FOLD, shuffle=True, random_state=CONFIG.RANDOM_STATE)

    for name, model in classifiers.items():
        scores = []
        for train_idx, valid_idx in kf.split(X):
            X_train, X_valid = X.iloc[train_idx], X.iloc[valid_idx]
            y_train, y_valid = y[train_idx], y[valid_idx]


            X_train_enc = preprocessor.fit_transform(X_train)
            X_valid_enc = preprocessor.transform(X_valid)

            X_train_enc, X_valid_enc = _scale(X_train_enc, X_valid_enc,
                                               list(range(X_train_enc.shape[1])))

            model.fit(X_train_enc, y_train)
            preds = model.predict(X_valid_enc)
            acc = accuracy_score(y_valid, preds)
            scores.append(acc)

        avg_score = np.mean(scores)
        print(f"{name}: {avg_score:.4f}")
        results.append({'Model': name, 'Accuracy': avg_score})
```

Scaling

```
def _scale(X_train, X_valid, features):
    scaler = StandardScaler()
    X_train_scaled = X_train.copy()
    X_valid_scaled = X_valid.copy()

    X_train_scaled[features] = scaler.fit_transform(X_train[features])
    X_valid_scaled[features] = scaler.transform(X_valid[features])

    return X_train_scaled, X_valid_scaled
```

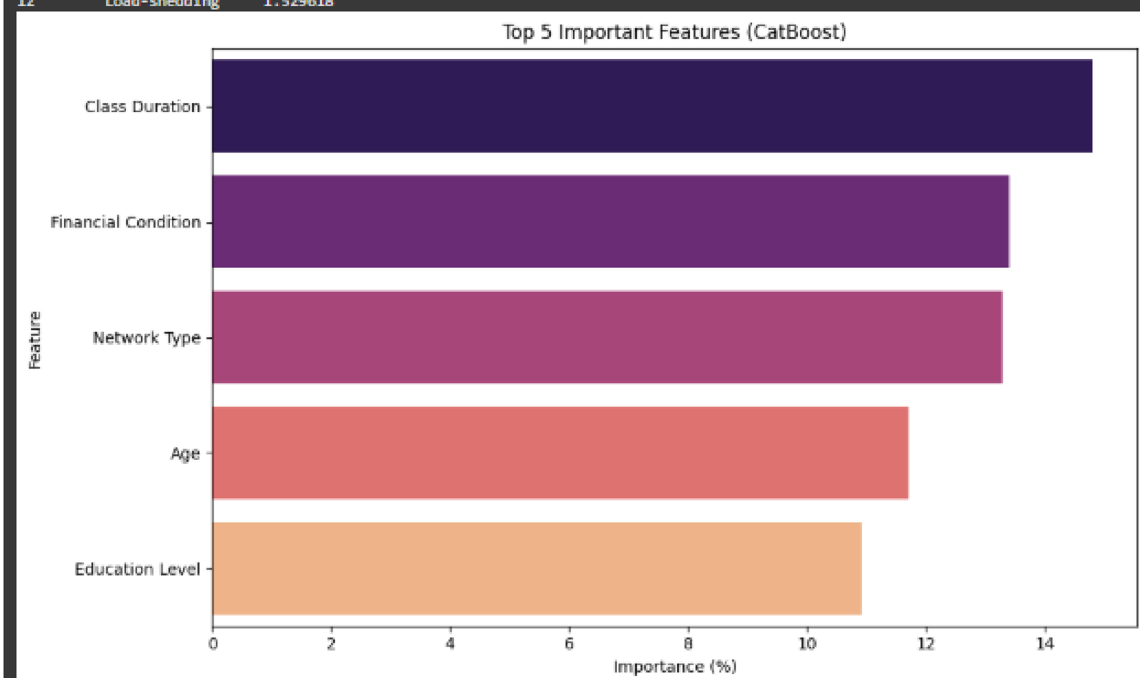
Milestone 4: Model Building

```python
import optuna
from xgboost import XGBClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score

def xgb_objective(trial):
    param = {
        'n_estimators': trial.suggest_int('n_estimators', 100, 1000),
        'max_depth': trial.suggest_int('max_depth', 3, 20),
        'reg_alpha': trial.suggest_int('reg_alpha', 0, 5),
        'reg_lambda': trial.suggest_int('reg_lambda', 0, 5),
        'min_child_weight': trial.suggest_int('min_child_weight', 0, 5),
        'gamma': trial.suggest_float('gamma', 0, 5),
        'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.3, log=True),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.3, 1.0),
        'n_jobs': -1,
        'random_state': CONFIG.RANDOM_STATE,
        'use_label_encoder': False,
        'eval_metric': 'mlogloss'  # For multi-class stability
    }

    accuracy = 0
    kf = StratifiedKFold(n_splits=CONFIG.FOLD, shuffle=True, random_state=CONFIG.RANDOM_STATE)

    for train_idx, val_idx in kf.split(X_scaled, y_encoded):
        x_train, x_val = X_scaled[train_idx], X_scaled[val_idx]
        y_train, y_val = y_encoded[train_idx], y_encoded[val_idx]

        try:
            model = XGBClassifier(**param)
            model.fit(x_train, y_train)
            val_preds = model.predict(x_val)
            accuracy += accuracy_score(y_val, val_preds) / CONFIG.FOLD
```

```
[I 2025-07-04 15:44:27,973] A new study created in memory with name: no-name-1db308da-1368-4383-a94e-7da7f7fb6dfc
[I 2025-07-04 15:44:35,018] Trial 0 finished with value: 0.8954356846473028 and parameters: {'iterations': 519, 'depth': 5, 'learning_rate': 0.10585561215732568, 'random_strength': 56
[I 2025-07-04 15:44:42,967] Trial 1 finished with value: 0.9128630705394192 and parameters: {'iterations': 472, 'depth': 5, 'learning_rate': 0.14392777995160735, 'random_strength': 72
[I 2025-07-04 15:45:22,811] Trial 2 finished with value: 0.912033195020747 and parameters: {'iterations': 489, 'depth': 8, 'learning_rate': 0.09618108015624052, 'random_strength': 34,
[I 2025-07-04 15:45:33,153] Trial 3 finished with value: 0.7643153526970954 and parameters: {'iterations': 203, 'depth': 8, 'learning_rate': 0.012330415327824695, 'random_strength': 3
[I 2025-07-04 15:46:13,939] Trial 4 finished with value: 0.8572614107883818 and parameters: {'iterations': 709, 'depth': 8, 'learning_rate': 0.01766706793863649б, 'random_strength': 5
[I 2025-07-04 15:46:39,505] Trial 5 finished with value: 0.9112033195020748 and parameters: {'iterations': 541, 'depth': 7, 'learning_rate': 0.0681939183000495, 'random_strength': 18,
[I 2025-07-04 15:46:58,288] Trial 6 finished with value: 0.9095435684647304 and parameters: {'iterations': 745, 'depth': 6, 'learning_rate': 0.08459285173248925, 'random_strength': 7,
[I 2025-07-04 15:47:23,854] Trial 7 finished with value: 0.910373443983402б and parameters: {'iterations': 333, 'depth': 8, 'learning_rate': 0.09732538755689805, 'random_strength': 58
[I 2025-07-04 15:47:35,189] Trial 8 finished with value: 0.910373443983402б and parameters: {'iterations': 742, 'depth': 5, 'learning_rate': 0.07992216656905463, 'random_strength': 2
[I 2025-07-04 15:47:47,486] Trial 9 finished with value: 0.910373443983402б and parameters: {'iterations': 478, 'depth': 6, 'learning_rate': 0.08761234933932856, 'random_strength': 42
[I 2025-07-04 15:47:50,521] Trial 10 finished with value: 0.8066390041493777 and parameters: {'iterations': 121, 'depth': 6, 'learning_rate': 0.03677659014083848, 'random_strength': 7
[I 2025-07-04 15:48:46,452] Trial 11 finished with value: 0.907883817427386 and parameters: {'iterations': 355, 'depth': 9, 'learning_rate': 0.141414077424893l, 'random_strength': 80,
[I 2025-07-04 15:50:18,454] Trial 12 finished with value: 0.912033195020747 and parameters: {'iterations': 634, 'depth': 9, 'learning_rate': 0.04867589359085882, 'random_strength': 64
[I 2025-07-04 15:50:35,523] Trial 13 finished with value: 0.9095435684647304 and parameters: {'iterations': 370, 'depth': 7, 'learning_rate': 0.14918774855266853, 'random_strength': 3
[I 2025-07-04 15:51:21,696] Trial 14 finished with value: 0.895435684647303 and parameters: {'iterations': 582, 'depth': 8, 'learning_rate': 0.0315398746778988 2, 'random_strength': 28
[I 2025-07-04 15:51:27,232] Trial 15 finished with value: 0.8556016597510373 and parameters: {'iterations': 423, 'depth': 5, 'learning_rate': 0.05533048697083661, 'random_strength': 6
[I 2025-07-04 15:51:52,659] Trial 16 finished with value: 0.866390041493776 and parameters: {'iterations': 631, 'depth': 7, 'learning_rate': 0.02665048214317775, 'random_strength': 45
[I 2025-07-04 15:52:04,691] Trial 17 finished with value: 0.9070539419087138 and parameters: {'iterations': 257, 'depth': 7, 'learning_rate': 0.12382330829513698, 'random_strength': 6
[I 2025-07-04 15:52:16,442] Trial 18 finished with value: 0.8829875518672199 and parameters: {'iterations': 437, 'depth': 6, 'learning_rate': 0.05442002613049973, 'random_strength': 2
[I 2025-07-04 15:53:02,217] Trial 19 finished with value: 0.912033195020747 and parameters: {'iterations': 280, 'depth': 9, 'learning_rate': 0.11643636859373215, 'random_strength': 69
[I 2025-07-04 15:53:52,391] Trial 20 finished with value: 0.9112033195020748 and parameters: {'iterations': 626, 'depth': 8, 'learning_rate': 0.06770708532762822, 'random_strength': 5
[I 2025-07-04 15:55:13,878] Trial 21 finished with value: 0.9136929460580914 and parameters: {'iterations': 570, 'depth': 9, 'learning_rate': 0.04829016967980194, 'random_strength': 6
✅ Best Accuracy Score: 0.9136929460580914
🥇 Best Hyperparameters: {'iterations': 570, 'depth': 9, 'learning_rate': 0.04829016967980194, 'random_strength': 63, 'bagging_temperature': 1.9644230050947629, 'od_type': 'IncToDec'
```

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

feature_importances = final_model.get_feature_importance(prettified=True)


feature_importances = feature_importances.sort_values(by='Importances', ascending=False)


print(feature_importances)

plt.figure(figsize=(10, 6))
sns.barplot(
    x='Importances',
    y='Feature Id',
    data=feature_importances.head(5),
    palette='magma'
)
plt.title('Top 5 Important Features (CatBoost)')
plt.xlabel('Importance (%)')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

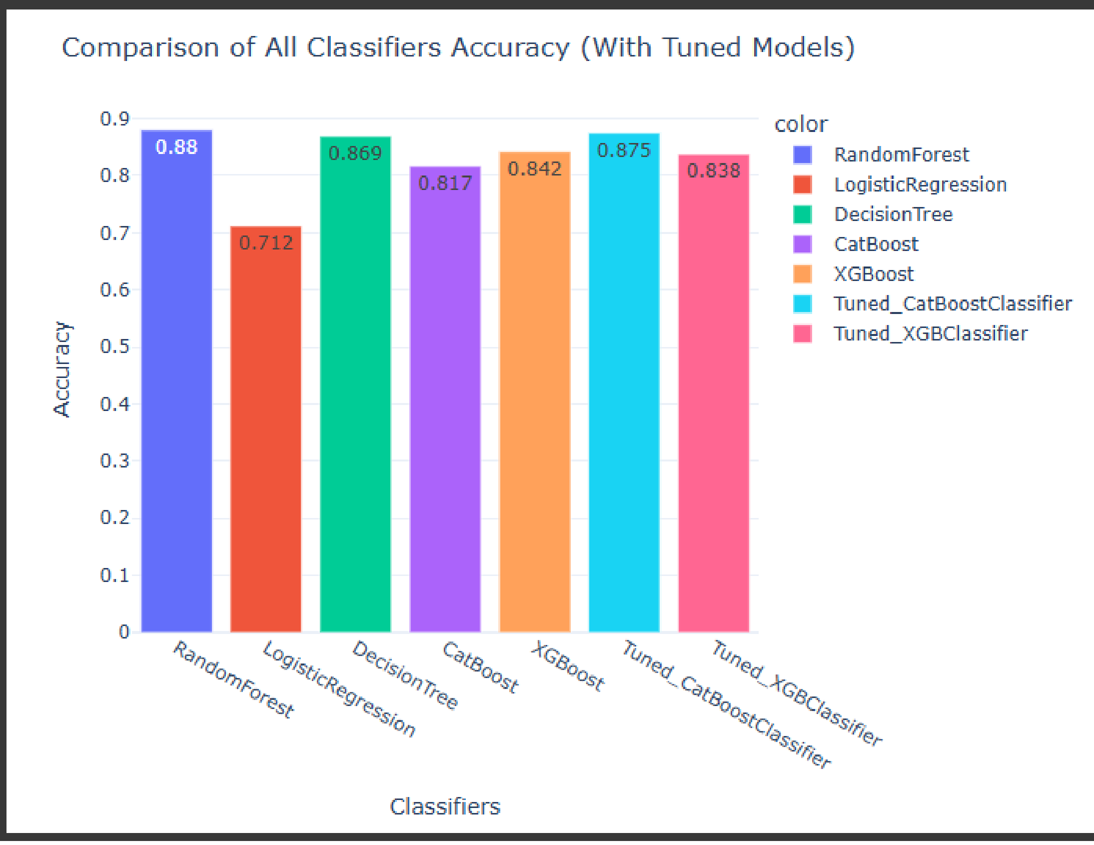
```
             Feature Id  Importances
0        Class Duration    14.812383
1   Financial Condition    13.391358
2          Network Type    13.293664
3                   Age    11.711763
4       Education Level    10.910286
5                Device     9.124106
6                Gender     9.014186
7      Institution Type     4.631611
8         Internet Type     4.427017
9              Location     2.843581
10             Self Lms     2.523321
11           IT Student     1.787106
12         Load-shedding     1.529618
```



Milestone 5: Performance Testing & Hyperparameter Tuning

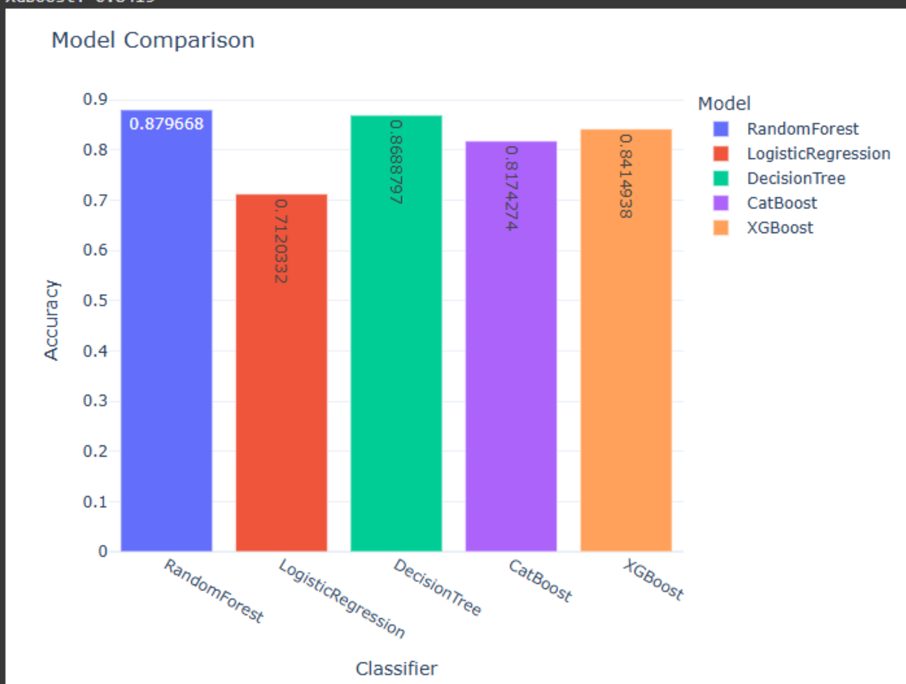
Activity 1.1: Compare the model

```python
p_bar_plot(
    x=classifiers_name,
    y=np.round(classifiers_accuracy, 3),
    width=700,
    x_title='Classifiers',
    y_title='Accuracy',
    title='Comparison of All Classifiers Accuracy (With Tuned Models)'
)
```



```python
results = classifiers_modeling(CONFIG.CLASSIFIERS, X, y, CONFIG.FEATURES)

results_df = pd.DataFrame(results)

p_bar_plot(results_df, x='Model', y='Accuracy', width=700,
           xlabel='Classifier', ylabel='Accuracy', title='Model Comparison')
```

```
RandomForest: 0.8797
LogisticRegression: 0.7120
DecisionTree: 0.8689
CatBoost: 0.8174
XGBoost: 0.8415
```



Tuned Catboost:

```python
accuracy = accuracy_score(y_test, predictions)
report = classification_report(y_test, predictions)
conf_matrix = confusion_matrix(y_test, predictions)

print("✅ Accuracy:", accuracy)
print("\n📋 Classification Report:\n", report)
print("\n🧮 Confusion Matrix:\n", conf_matrix)


results = {
    'model': final_model,
    'predictions': predictions,
    'accuracy': accuracy,
    'classification_report': report,
    'confusion_matrix': conf_matrix
}
```

```
✅ Accuracy: 0.8838174273858921

📋 Classification Report:
               precision    recall  f1-score   support

        High       0.81      0.57      0.67        23
         Low       0.92      0.90      0.91       103
    Moderate       0.86      0.93      0.90       115

    accuracy                           0.88       241
   macro avg       0.87      0.80      0.82       241
weighted avg       0.88      0.88      0.88       241


🧮 Confusion Matrix:
 [[ 13   0  10]
 [  3  93   7]
 [  0   8 107]]
```

Milestone 6: Model Deployment

```python
def manual_transform(user_input_dict):
    encoded = []
    for feature in FEATURES:  # Use fixed order
        value = user_input_dict.get(feature)
        if value is None:
            raise KeyError(f"Feature '{feature}' missing in input")
        if feature in encoder_mapping:
            try:
                encoded_val = encoder_mapping[feature].index(value)
            except ValueError:
                raise ValueError(f"Invalid value '{value}' for feature '{feature}'")
            encoded.append(encoded_val)
        else:
            raise KeyError(f"Feature '{feature}' not found in mapping")
    return [np.array(encoded, dtype=int)]



model = CatBoostClassifier()
model.load_model("catboost_retrained_model.cbm")

@app.route('/')
def home():
    return render_template('home.html')
```

```python
from flask import Flask, request, render_template
from catboost import CatBoostClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import pandas as pd
import numpy as np
import json

```

```python
# Features and categories from training
FEATURES = [
    'Gender', 'Age', 'Education Level', 'Institution Type', 'IT Student',
    'Location', 'Load-shedding', 'Financial Condition', 'Internet Type',
    'Network Type', 'Class Duration', 'Self Lms', 'Device'
]

CATEGORIES = [
    ['Boy', 'Girl'],
    ['11-15', '16-20', '21-25'],
    ['School', 'College', 'University'],
    ['Government', 'Non Government'],
    ['Yes', 'No'],  # IT Student
    ['Yes', 'No'],  # Location
    ['High', 'Low'],
    ['Poor', 'Mid', 'Rich'],
    ['Mobile Data', 'Wifi'],
    ['2G', '3G', '4G', 'No Internet Service'],
    ['0', '1-3', '3-6'],
    ['Yes', 'No'],
    ['Mobile', 'Laptop', 'Tab']
]

```

```python

    # ✅ Directly convert to DataFrame with raw string features
    input_df = pd.DataFrame([user_input])

    # 🧠 Predict
    prediction = model.predict(input_df)
    probabilities = model.predict_proba(input_df)

    print("Prediction:", prediction)
    print("Probabilities:", probabilities)

    return render_template('result.html', prediction=prediction[0])

if __name__ == '__main__':
    app.run(debug=True)

```

```python
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/form')
def form():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    user_input = {
        "Gender": request.form['gender'],
        "Age": request.form['age'],
        "Education Level": request.form['education_level'],
        "Institution Type": request.form['institute_type'],
        "IT Student": request.form['it_student'],
        "Location": request.form['location'],
        "Load-shedding": request.form['load_shedding'],
        "Financial Condition": request.form['financial_condition'],
        "Internet Type": request.form['internet_type'],
        "Network Type": request.form['network_type'],
        "Class Duration": request.form['class_duration'],
        "Self Lms": request.form['self_lms'],
        "Device": request.form['device']
    }
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Adaptability Prediction</title>
    <style>
        body { font-family: Arial; padding: 20px; }
        label { display: block; margin: 10px 0 5px; }
        input, select { width: 300px; padding: 5px; }
        .submit-btn { margin-top: 20px; padding: 10px 20px; }
    </style>
</head>
<body>
    <h2>Enter Details for Adaptability Prediction</h2>
    <form action="/predict" method="post">

        <label>Gender:</label>
        <select name="gender" required>
            <option value="Boy">Boy</option>
            <option value="Girl">Girl</option>
        </select>

        <label>Age:</label>
        <select name="age" required>
            <option value="1-5">1-5</option>
            <option value="6-10">6-10</option>
            <option value="11-15">11-15</option>
```

```python

    # ✅ Directly convert to DataFrame with raw string features
    input_df = pd.DataFrame([user_input])

    # 🧠 Predict
    prediction = model.predict(input_df)
    probabilities = model.predict_proba(input_df)

    print("Prediction:", prediction)
    print("Probabilities:", probabilities)

    return render_template('result.html', prediction=prediction[0])

if __name__ == '__main__':
    app.run(debug=True)

```

```html
            <option value="16-20">16-20</option>
            <option value="21-25">21-25</option>
            <option value="26-30">26-30</option>
        </select>

        <label>Education Level:</label>
        <select name="education_level" required>
            <option value="School">School</option>
            <option value="College">College</option>
            <option value="University">University</option>
        </select>

        <label>Institution Type:</label>
        <select name="institute_type" required>
            <option value="Government">Government</option>
            <option value="Non Government">Non Government</option>
        </select>

        <label>IT Student:</label>
        <select name="it_student" required>
            <option value="No">No</option>
            <option value="Yes">Yes</option>
```

Enter Details for Adaptability Prediction

Gender:

Boy

Age:

1-5

Education Level:

School

Institution Type:

Government

IT Student:

No

Location:

No

Load Shedding:

Low

Financial Condition:

Poor

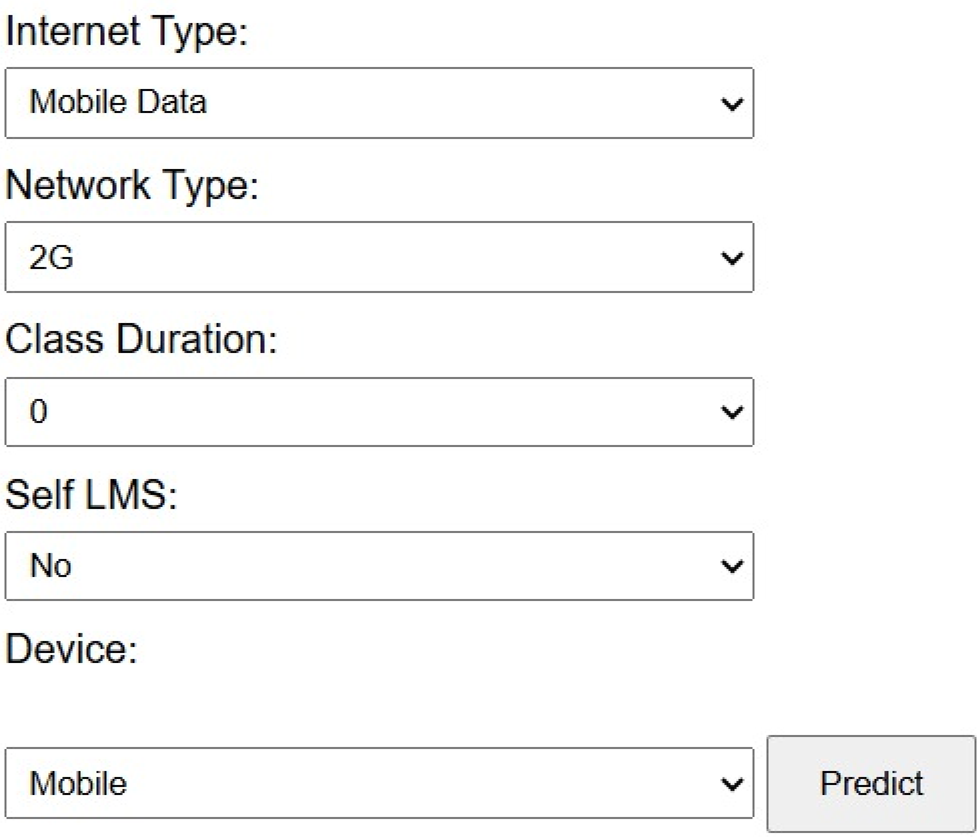
Internet Type:

Mobile Data

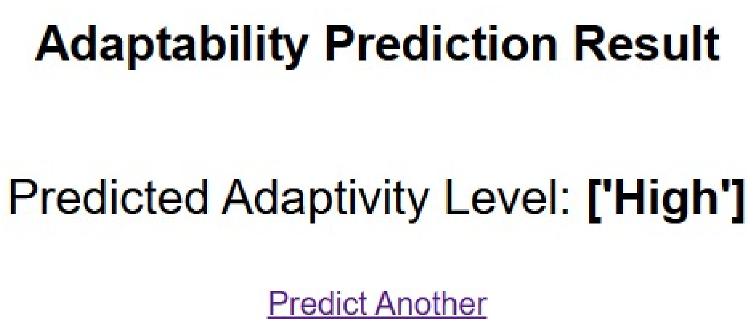
Network Type:

2G

Class Duration:


Internet Type:
Mobile Data
Network Type:
2G
Class Duration:
0
Self LMS:
No
Device:
Mobile
Predict

Prediction


Adaptability Prediction Result
Predicted Adaptivity Level: ['High']
Predict Another