E-Adapt: Predicting Student Adaptability In Online Classes

1.INTRODUCTION

1.1. OVERVIEW

"E-Adapt: Predicting Student Adaptability In Online Classes" seems to be a project focused on developing a system or model to predict student adaptability in the context of online classes. The term "E-Adapt" suggests that it may be related to e-learning or electronic learning environments.

1.2. PURPOSE

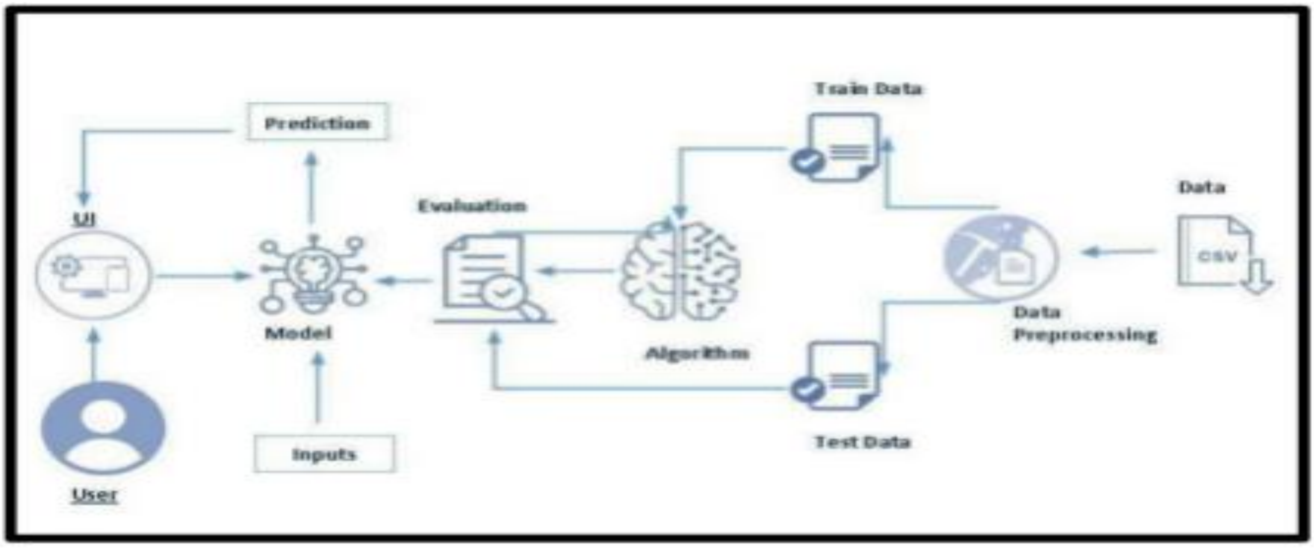
The purpose of the project is likely to address the challenges associated with online education and to enhance the overall learning experience for students. Predicting student adaptability could involve assessing how well students can adjust to the online learning environment, engage with digital tools, manage their time effectively, and overcome potential obstacles.

2.LITERATURE SURVEY

The literature survey on predicting student adaptability in online classes highlights the significance of adaptability for student success in digital learning environments. It explores factors influencing adaptability such as prior academic performance, technological proficiency, self-regulated learning skills, motivation, engagement, and personal characteristics. The survey examines existing approaches, ranging from traditional statistical models to machine learning algorithms, and discusses relevant data sources for prediction. It emphasizes the need for accurate prediction models to enhance student outcomes and improve the effectiveness of online education.

3.THEORITICAL ANALYSIS

3.1.BLOCK DIAGRAM



3.2 **HARDWARE / SOFTWARE DESIGNING**

Hardware Design:

1.Data Collection Devices:

Laptops, desktops, or tablets used by students for online classes.Sensors (if applicable) to collect additional data, such as eye-tracking devices or biometric sensors to measure stress levels.

2.Server Infrastructure:

Servers for storing and processing collected data.Sufficient processing power and memory to handle predictive modeling tasks.

3.Networking:

Reliable and high-speed internet connectivity to ensure seamless data transfer between student devices and servers.

Software Design:

1.Data Collection Software:

Applications or scripts running on student devices to collect data.Browser extensions or plugins to gather information related to online behavior.

2.Database Management:

Database systems (e.g., MySQL, PostgreSQL) to store collected data securely.Data cleaning and preprocessing tools to ensure the quality of the dataset.

3.Predictive Modeling Software:

Machine learning frameworks (e.g., TensorFlow, PyTorch) for building predictive models.Statistical analysis tools for identifying key factors influencing adaptability.

4.Programming Languages:

Python is commonly used for machine learning tasks. Other languages may be used for specific components.

5.Web Development :

Development of a web-based interface for users (students, instructors) to interact with the system. Dashboards to visualize predictions and insights.

6.User Authentication and Security:

Implement secure user authentication mechanisms to protect sensitive student data.Encryption protocols to ensure data privacy.

7.Model Deployment:

Deployment tools and frameworks (e.g., Docker, Kubernetes) for deploying machine learning models.Integration with the online learning platform or educational systems.

8.Monitoring and Logging:

Tools for monitoring the system's performance and logging events for troubleshooting.Alerts for unusual activities or system failures.

9.Documentation and Version Control:

Comprehensive documentation for both hardware and software components.Version control systems (e.g., Git) for tracking changes in the codebase.

10.Ethical Considerations:

Implementation of ethical guidelines for handling student data.Compliance with data protection regulations (e.g., GDPR)

4.EXPERIMENTAL INVESTIGATIONS

Project Flow:

• User is shown the Home page. The user will browse through Home page and go to predict my adaptivity and enter the specified engagement metrics.

• After clicking the Predict button the user will be directed to the Results page where the model will analyse the inputs given by the user and showcase the prediction of the Adaptivity level.

To accomplish this we have to complete all the activities listed below:

• Define problem / Problem understanding
o Specify the business problem
o Business Requirements
o Literature Survey
o Social or Business Impact
• Data Collection and Preparation
o Collect the dataset
o Data Preparation
• Exploratory Data Analysis
o Descriptive statistical
o Visual Analysis
• Model Building
o Creating a function for evaluation
o Training and testing the Models using multiple algorithms
• Performance Testing & Hyperparameter Tuning
o Testing model with multiple evaluation metrics
o Comparing model accuracy before & after applying hyperparameter tuning o Comparing model accuracy for different number of features.
o Building model with appropriate features.
• Model Deployment
o Save the best model
o Integrate with Web Framework

5. RESULT

Activity 1: Collect The Dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.
Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 2: Import Necessary Libraries

```
# (1) Data
import numpy as np
import pandas as pd
# (2) Visualization
import seaborn as sns
import matplotlib as mpl
import plotly.express as px
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
# (3) Preprocessing
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
# (4) Modeling Libraries
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
# (5) Tuning Libraries
import optuna
# (6) Metrics & Scoring Libraries
from sklearn.model_selection import cross_val_score
# from sklearn.metrics import confusion_matrix, plot_confusion_matrix, classification_report
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# (7) Options
%matplotlib inline
from termcolor import colored
plt.rcParams['axes.unicode_minus'] = False
pd.reset_option('display.float_format')
pd.set_option('display.max_columns', None)
color_scheme = px.colors.qualitative.Pastel
```

Activity 3: Read The Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

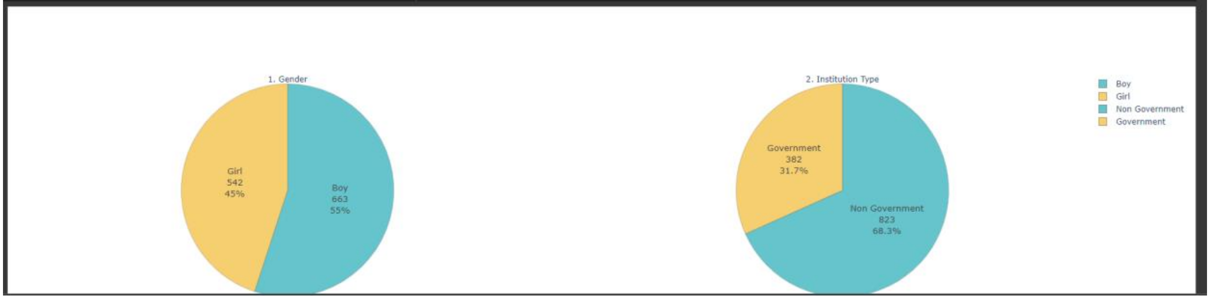
```
[ ] df = pd.read_csv("/content/drive/MyDrive/students_adaptability_level_online_education.csv")

    print(colored('Shape of DataFrame: ','blue'), df.shape, '\n\n')
    df.head()
```

Shape of DataFrame: (1205, 14)

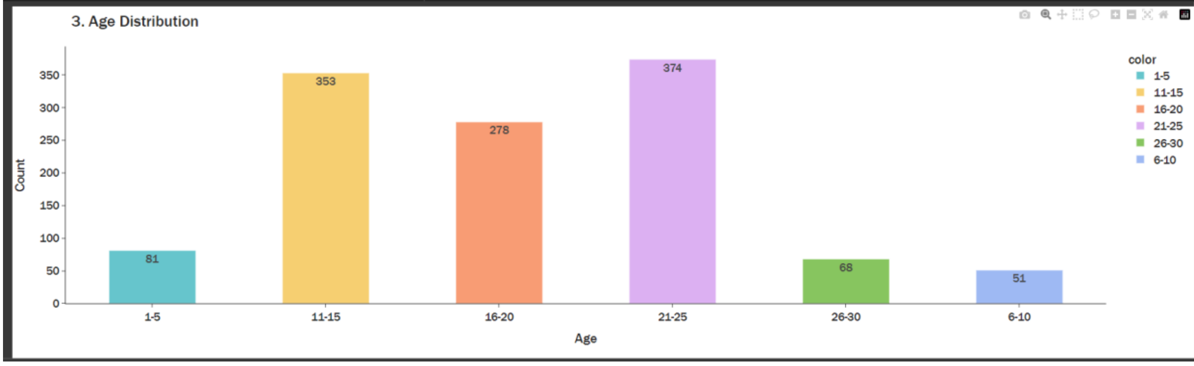
	Gender	Age	Education Level	Institution Type	IT Student	Location	Load-shedding	Financial Condition	Internet Type	Network Type	Class Duration	Self Lms	Device	Adaptivity Level
0	Boy	21-25	University	Non Government	No	Yes	Low	Mid	Wifi	4G	3-6	No	Tab	Moderate
1	Girl	21-25	University	Non Government	No	Yes	High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

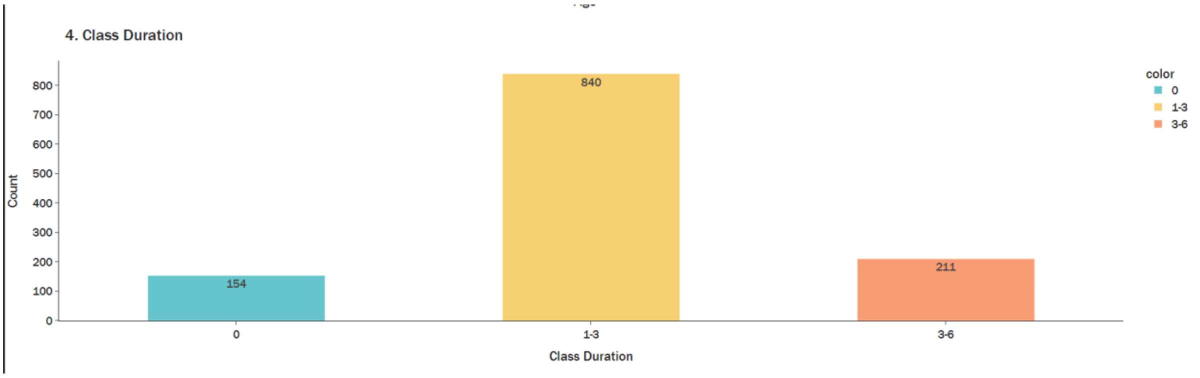
Activity 4: Univariate Analysis

```
series_1 = []
series_1_title = ['Gender', 'Institution Type']
series_1.append(df['Gender'].value_counts())
series_1.append(df['Institution Type'].value_counts())

fig = make_subplots(rows=1, cols= 2, specs =[[{"type": "pie"},{"type":"pie"}]])
for idx, series in enumerate(series_1):
    fig.add_trace(go.Pie(values = series.values,
                    labels = series.index,
                    marker = dict(colors = color_scheme),
                    title = str(idx+1)+'. '+series_1_title[idx]),
                    row = 1,
                    col = idx + 1)
    fig.update_traces(textinfo='label+percent+value', textfont_size=13,
    marker=dict(line=dict(color='#100000', width=0.2)))
fig.show()
```

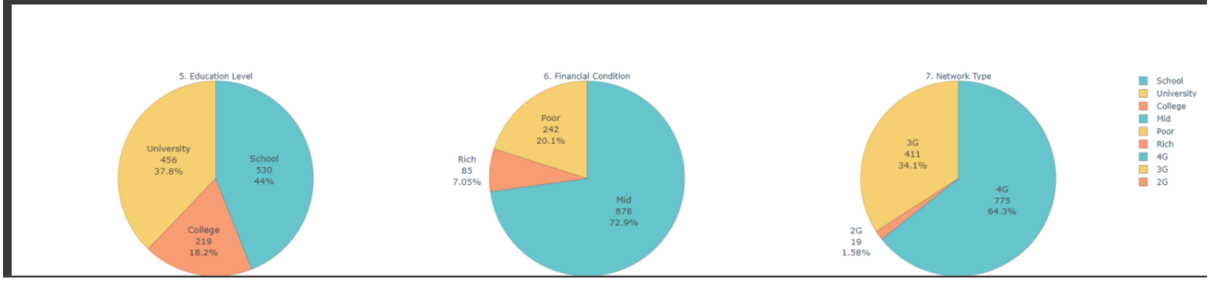



```
age = df['Age'].value_counts().sort_index()
class_duration = df['Class Duration'].value_counts().sort_index()
p_bar_plot(age.index, age.values, 'Age', 'Count', '3. Age Distribution')
p_bar_plot(class_duration.index, class_duration.values, 'Class Duration', 'Count', '4. Class Duration')
```

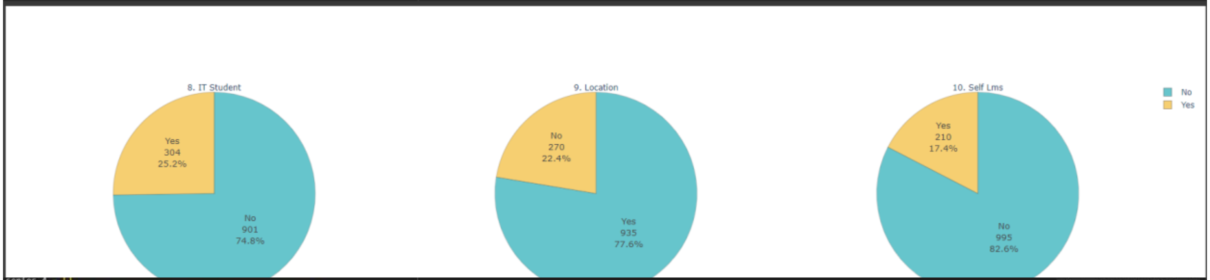




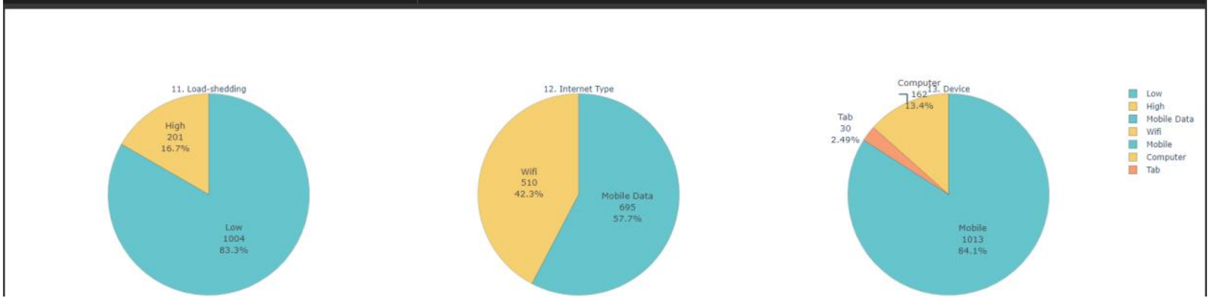
```python
series_2 = []
series_2_title = ['Education Level', 'Financial Condition', 'Network Type']
series_2.append(df['Education Level'].value_counts())
series_2.append(df['Financial Condition'].value_counts())
series_2.append(df['Network Type'].value_counts())

fig = make_subplots(rows=1, cols= 3, specs =[[{"type": "pie"},{"type":"pie"},{"type":"pie"}]])
for idx, series in enumerate(series_2):
    fig.add_trace(go.Pie(values = series.values,
                         labels = series.index,
                         marker = dict(colors = color_scheme),
                         title = str(idx+5)+'. '+series_2_title[idx]),
                         row = 1,
                         col = idx + 1)
    fig.update_traces(textinfo='label+percent+value', textfont_size=13,
    marker=dict(line=dict(color='#100000', width=0.2)))
fig.show()
```



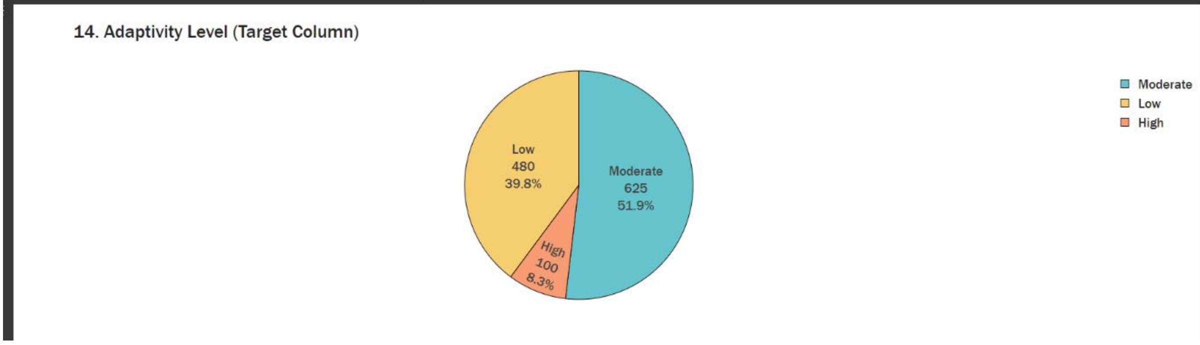

```python
series_3 = []
series_3_title = ['IT Student', 'Location', 'Self Lms']
series_3.append(df['IT Student'].value_counts())
series_3.append(df['Location'].value_counts())
series_3.append(df['Self Lms'].value_counts())

fig = make_subplots(rows=1, cols= 3, specs =[[{"type": "pie"},{"type":"pie"},{"type":"pie"}]])
for idx, series in enumerate(series_3):
    fig.add_trace(go.Pie(values = series.values,
                         labels = series.index,
                         marker = dict(colors = color_scheme),
                         title = str(idx+8)+'. '+series_3_title[idx]),
                         row = 1,
                         col = idx + 1)
    fig.update_traces(textinfo='label+percent+value', textfont_size=13,
    marker=dict(line=dict(color='#100000', width=0.2)))
fig.show()
```




```python
series_4 = []
series_4_title = ['Load-shedding', 'Internet Type', 'Device']
series_4.append(df['Load-shedding'].value_counts())
series_4.append(df['Internet Type'].value_counts())
series_4.append(df['Device'].value_counts())
fig = make_subplots(rows=1, cols= 3, specs =[[{"type": "pie"},{"type":"pie"},{"type":"pie"}]])
for idx, series in enumerate(series_4):
    fig.add_trace(go.Pie(values = series.values,
                         labels = series.index,
                         marker = dict(colors = color_scheme),
                         title = str(idx+11)+'. '+series_4_title[idx]),
                         row = 1,
                         col = idx + 1)
    fig.update_traces(textinfo='label+percent+value', textfont_size=13,
    marker=dict(line=dict(color='#100000', width=0.2)))
fig.show()
```

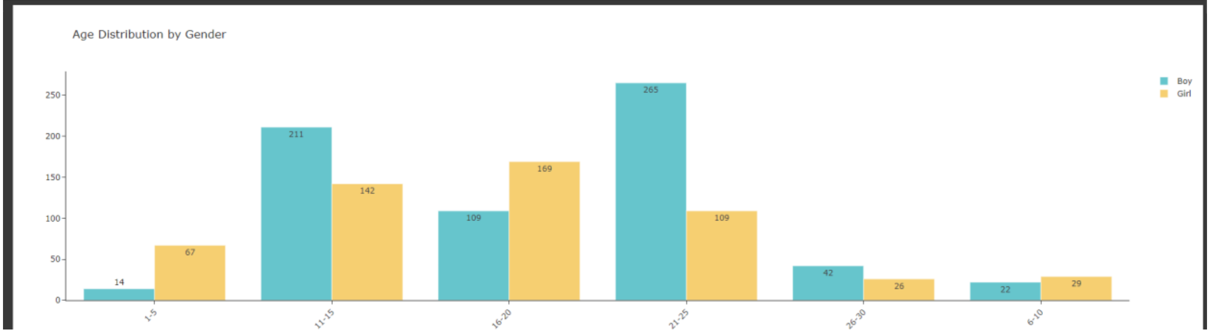


```
p_pie_plot(df['Adaptivity Level'].value_counts(), '14. Adaptivity Level (Target Column)')
```

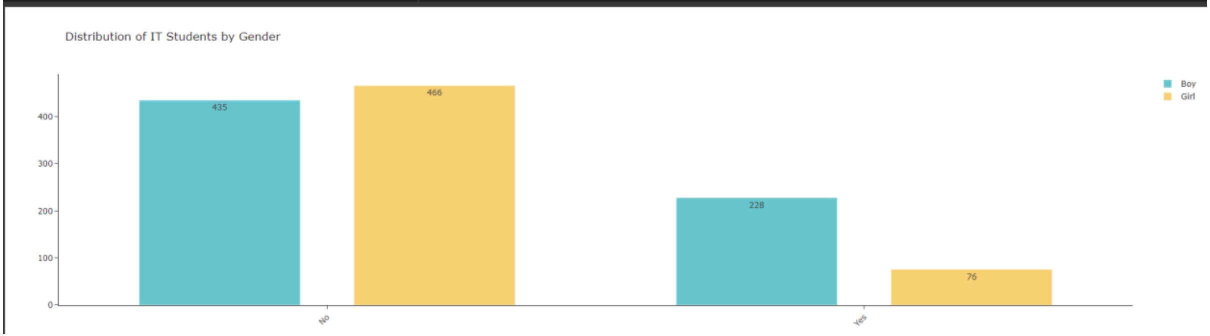


Activity 5: Multivariate Analysis

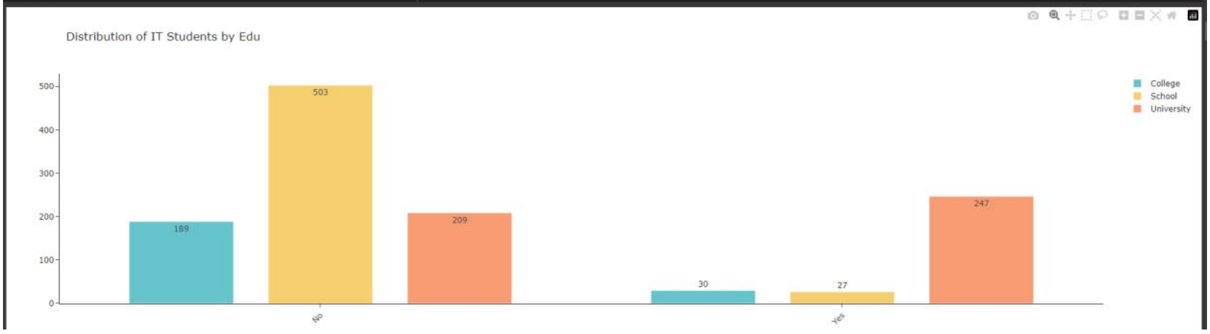
```
gender_age = df.groupby(['Gender','Age']).size()
fig = go.Figure(data=[
    go.Bar(name='Boy', x=gender_age['Boy'].index, y=gender_age['Boy'].values,
           text=gender_age['Boy'].values, marker_color=color_scheme[0]),

    go.Bar(name='Girl', x=gender_age['Girl'].index, y=gender_age['Girl'].values,
           text=gender_age['Girl'].values, marker_color=color_scheme[1])
])
fig.update_layout(barmode='group', xaxis_tickangle=-45,title='Age Distribution by Gender',
                  template = 'simple_white')

fig.show()
```



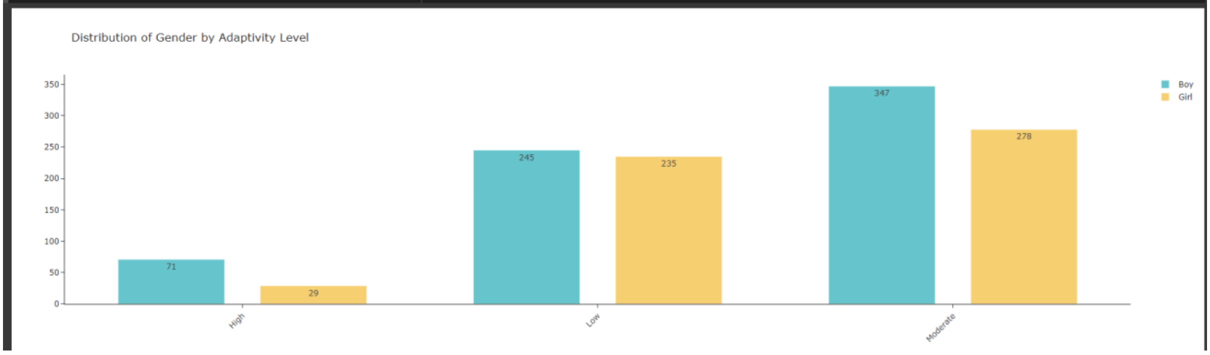

```
gender_IT = df.groupby(['Gender','IT Student']).size()
fig = go.Figure(data=[
    go.Bar(name='Boy', x=gender_IT['Boy'].index, y=gender_IT['Boy'].values,
           text=gender_IT['Boy'].values, marker_color=color_scheme[0]),

    go.Bar(name='Girl', x=gender_IT['Girl'].index, y=gender_IT['Girl'].values,
           text=gender_IT['Girl'].values, marker_color=color_scheme[1])
])
fig.update_layout(barmode='group', xaxis_tickangle=-45,title='Distribution of IT Students by Gender',
                  template = 'simple_white')
fig.update_traces(width=0.3)
fig.show()
```



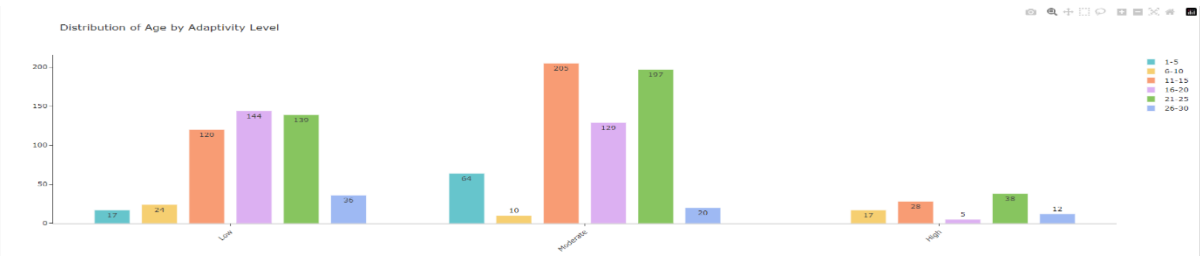
```python
Education_IT = df.groupby(['Education Level','IT Student']).size()
fig = go.Figure(data=[
    go.Bar(name='College', x=Education_IT['College'].index, y=Education_IT['College'].values,
           text=Education_IT['College'].values, marker_color=color_scheme[0]),
    go.Bar(name='School', x=Education_IT['School'].index, y=Education_IT['School'].values,
           text=Education_IT['School'].values, marker_color=color_scheme[1]),
    go.Bar(name='University', x=Education_IT['University'].index, y=Education_IT['University'].values,
           text=Education_IT['University'].values, marker_color=color_scheme[2])
])
fig.update_layout(barmode='group', xaxis_tickangle=-45,title='Distribution of IT Students by Edu',
                  template = 'simple_white')
fig.update_traces(width=0.2)
fig.show()
```




```python
Gender_Adaptivity = df.groupby(['Gender','Adaptivity Level']).size()
fig = go.Figure(data=[
    go.Bar(name='Boy', x=Gender_Adaptivity['Boy'].index, y=Gender_Adaptivity['Boy'].values,
           text=Gender_Adaptivity['Boy'].values, marker_color=color_scheme[0]),
    go.Bar(name='Girl', x=Gender_Adaptivity['Girl'].index, y=Gender_Adaptivity['Girl'].values,
           text=Gender_Adaptivity['Girl'].values, marker_color=color_scheme[1]),
])
fig.update_layout(barmode='group', xaxis_tickangle=-45,title='Distribution of Gender by Adaptivity Level',
                  template = 'simple_white')
fig.update_traces(width=0.3)
fig.show()
```




```python
Age_Adaptivity = df.groupby(['Age','Adaptivity Level']).size()
fig = go.Figure(data=[
    go.Bar(name='1-5', x=Age_Adaptivity['1-5'].index, y=Age_Adaptivity['1-5'].values,
           text=Age_Adaptivity['1-5'].values, marker_color=color_scheme[0]),

    go.Bar(name='6-10', x=Age_Adaptivity['6-10'].index, y=Age_Adaptivity['6-10'].values,
           text=Age_Adaptivity['6-10'].values, marker_color=color_scheme[1]),

    go.Bar(name='11-15', x=Age_Adaptivity['11-15'].index, y=Age_Adaptivity['11-15'].values,
           text=Age_Adaptivity['11-15'].values, marker_color=color_scheme[2]),

    go.Bar(name='16-20', x=Age_Adaptivity['16-20'].index, y=Age_Adaptivity['16-20'].values,
           text=Age_Adaptivity['16-20'].values, marker_color=color_scheme[3]),

    go.Bar(name='21-25', x=Age_Adaptivity['21-25'].index, y=Age_Adaptivity['21-25'].values,
           text=Age_Adaptivity['21-25'].values, marker_color=color_scheme[4]),

    go.Bar(name='26-30', x=Age_Adaptivity['26-30'].index, y=Age_Adaptivity['26-30'].values,
           text=Age_Adaptivity['26-30'].values, marker_color=color_scheme[5]),
])
fig.update_layout(barmode='group', xaxis_tickangle=-45,title='Distribution of Age by Adaptivity Level',
                  template = 'simple_white')

fig.update_traces(width=0.1)

fig.show()
```



Activity 6: Pre Processing

```
# df.columns.tolist()
'''
features:
['Gender','Age','Education Level','Institution Type','IT Student','Location',
 'Load-shedding','Financial Condition','Internet Type','Network Type','Class Duration','Self Lms','Device','Adaptivity Level']
'''
columns = ['Gender','Age','Education Level','Institution Type','IT Student','Location',
           'Load-shedding','Financial Condition','Internet Type','Network Type',
           'Class Duration','Self Lms','Device','Adaptivity Level']
```

Activity 7: Encoding

```
for column in columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
```

```
df.head(4)
```

	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	0	3	2	1	0	1	1	0	1	2	2	0	2	2
1	1	3	2	1	0	1	0	0	0	2	1	1	1	2
2	1	2	0	0	0	1	1	0	1	2	1	0	1	2
3	1	1	1	1	0	1	1	0	0	2	1	0	1	2

Activity 8: Train-Test Split

```
y = df['Adaptivity Level']
X = df.drop('Adaptivity Level', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 116)
```

```
print(colored('Shape of X_train: ','blue'), X_train.shape, '\n\n')

print(colored('Shape of X_test: ', 'red'), X_test.shape)
```

```
Shape of X_train:  (843, 13)


Shape of X_test:  (362, 13)
```

Activity 9: Define Pre-Processing And Modelling Function

```
class CONFIG:
    FOLD = 5                # Cross Validation Fold Num
    RANDOM_STATE = 116      # Random State
    # 4 models
    CLASSIFIERS = {
    "LogisiticRegression": LogisticRegression(max_iter = 1000,
                                              random_state = RANDOM_STATE),
    "KNearest": KNeighborsClassifier(),

    "RandomForestClassifier": RandomForestClassifier(random_state = RANDOM_STATE),

    "XGBClassifier": XGBClassifier(random_state = RANDOM_STATE),

    "CatBoostClassifier": CatBoostClassifier(random_state = RANDOM_STATE,
                                             logging_level='Silent')
    }
    # Features
    FEATURES = ['Gender','Age','Education Level','Institution Type','IT Student','Location',
           'Load-shedding','Financial Condition','Internet Type','Network Type',
           'Class Duration','Self Lms','Device']
    # CatBoost Best Parameters (using optuna)
    CB_BEST_PARAMS = {'iterations': 815, 'depth': 5,
                  'learning_rate': 0.08616774389778385,
                  'random_strength': 45, 'bagging_temperature': 0.23946457882908667,
                  'od_type': 'Iter', 'od_wait': 18,
                  'logging_level': 'Silent',
                  'random_state': RANDOM_STATE}
    # XGBoost Best Parameters (using optuna)
    XGB_BEST_PARAMS = {'n_estimators': 634, 'max_depth': 15,
                  'reg_alpha': 0, 'reg_lambda': 3,
                  'min_child_weight': 0, 'gamma': 0,
                  'learning_rate': 0.2616305059064822,
                  'colsample_bytree': 0.65,
                  'random_state': RANDOM_STATE}
```

```python
# scaler function
def _scale(train_data, val_data, features):
    scaler = StandardScaler()
    scaled_train = scaler.fit_transform(train_data[features])
    scaled_val = scaler.transform(val_data[features])

    train = train_data.copy()
    val = val_data.copy()

    train[features] = scaled_train
    val[features] = scaled_val

    return train, val
```

```python
def classifiers_modeling(classifiers, X_train, y_train_, X_test, y_test, features):
    accuracy_list = []
    classifiers_name = list(classifiers.keys())
    # 5 Fold
    fold = KFold(n_splits = CONFIG.FOLD, random_state = CONFIG.RANDOM_STATE, shuffle=True)
    for idx, classifier in enumerate(classifiers.values()):
        accuracy = 0
        for fold_idx, (train_idx, val_idx) in enumerate(fold.split(X_train, y_train_)):
            x_train, x_val = X_train.iloc[train_idx], X_train.iloc[val_idx]
            y_train, y_val = y_train_.iloc[train_idx], y_train_.iloc[val_idx]
            x_train, x_val = _scale(x_train, x_val, features)
            model = classifier.fit(x_train[features], y_train)
            val_preds = model.predict(x_val[features])
            accuracy += accuracy_score(y_val, val_preds) / CONFIG.FOLD
        accuracy_list.append(round(accuracy,5))

        print('(',idx+1,')', classifiers_name[idx], 'cross validation (5 fold)')
        print('Mean Accurcy Score: ', round(accuracy, 5))
#         print('Mean Accuracy Score: ', colored(round(accuracy,5)))
    return accuracy_list
```

```python
# Model Score Visualization
def p_bar_plot(x, y, width, x_axis_title, y_axis_title, title):
    fig = px.bar(x = x,
                 y = y,
                 color = x,
                 text = y,
                 color_discrete_sequence = color_scheme,
                 template = 'simple_white')

    fig.update_layout(xaxis_title = x_axis_title,
                      yaxis_title= y_axis_title,
                      title = title,
                      font = dict(size=17,family="Franklin Gothic"))

    fig.update_traces(width=width)

    fig.show()
```

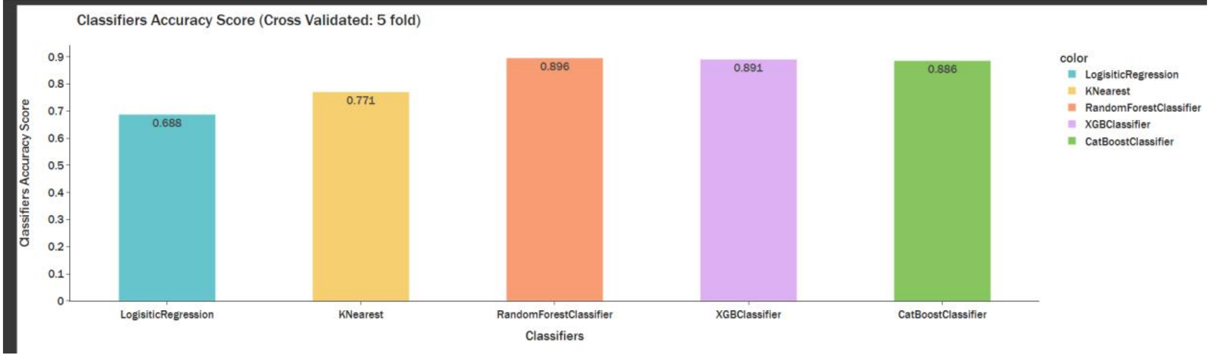
Activity 10: Model Comparision Part-1

```python
classifiers_name = list(CONFIG.CLASSIFIERS.keys())
classifiers_accuracy = []
```

```python
classifiers_accuracy = classifiers_modeling(CONFIG.CLASSIFIERS, X_train, y_train, X_test, y_test, CONFIG.FEATURES)
```

```
( 1 ) LogisiticRegression cross validation (5 fold)
Mean Accurcy Score:  0.68809
( 2 ) KNearest cross validation (5 fold)
Mean Accurcy Score:  0.77112
( 3 ) RandomForestClassifier cross validation (5 fold)
Mean Accurcy Score:  0.89568
( 4 ) XGBClassifier cross validation (5 fold)
Mean Accurcy Score:  0.89094
( 5 ) CatBoostClassifier cross validation (5 fold)
Mean Accurcy Score:  0.88619
```

```
p_bar_plot(x = classifiers_name,
           y = np.round(classifiers_accuracy,3),
           width = 0.5,
           x_axis_title = 'Classifiers',
           y_axis_title = 'Classifiers Accuracy Score',
           title = 'Classifiers Accuracy Score (Cross Validated: 5 fold)'
              )
```



Activity 11: CatBoost Classifier Hyperparameter Tuning

```
# Assuming you have imported or defined X_train and y_train
y_train_ = y_train
```

```
def cb_objective(trial):
    # CB Classifier Params
    params = {
        'iterations' : trial.suggest_int('iterations', 50, 1000),
        'depth' : trial.suggest_int('depth', 4, 10),
        'learning_rate' : trial.suggest_loguniform('learning_rate', 0.01, 0.3),
        'random_strength' :trial.suggest_int('random_strength', 0, 100),
        'bagging_temperature' :trial.suggest_loguniform('bagging_temperature', 0.01, 100.00),
        'od_type': trial.suggest_categorical('od_type', ['IncToDec', 'Iter']),
        'od_wait' :trial.suggest_int('od_wait', 10, 50),
        'logging_level': 'Silent',
        'random_state': CONFIG.RANDOM_STATE
    }
    # Set Accuracy to zero
    accuracy = 0
    # Create Model
    cb_model = CatBoostClassifier(**params)
    # 5 Fold
    fold = KFold(n_splits = CONFIG.FOLD, random_state = CONFIG.RANDOM_STATE, shuffle=True)
    for fold_idx, (train_idx, val_idx) in enumerate(fold.split(X_train, y_train_)):
        x_train, x_val = X_train.iloc[train_idx], X_train.iloc[val_idx]
        x_train, x_val = _scale(x_train, x_val, CONFIG.FEATURES)
        y_train, y_val = y_train_.iloc[train_idx], y_train_.iloc[val_idx]
        model = cb_model.fit(x_train[CONFIG.FEATURES], y_train)
        val_preds = model.predict(x_val[CONFIG.FEATURES])

        accuracy += accuracy_score(y_val, val_preds) / CONFIG.FOLD
    # return /5 fold accuracy score
    return accuracy
```

Activity12: XGB Classifier Hyperparameter Tuning

```
result = Times()
# Make sure X_train and CONFIG are defined before using them
# ...
y_train_ = y_train
def xgb_objective(trial):
    param = {
        "n_estimators": trial.suggest_int('n_estimators', 0, 1000),
        'max_depth': trial.suggest_int('max_depth', 2, 25),
        '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_int('gamma', 0, 5),
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.005, 0.5),
        'colsample_bytree': trial.suggest_discrete_uniform('colsample_bytree', 0.1, 1, 0.01),
        'nthread': -1
    }
    # Get Accuracy
    accuracy = 0
    # Create Model
    xgb_model = XGBClassifier(**param)
    # 5 Fold
    fold = KFold(n_splits=CONFIG.FOLD, random_state=CONFIG.RANDOM_STATE, shuffle=True)
    for fold_idx, (train_idx, val_idx) in enumerate(fold.split(X_train, y_train_)):
        x_train, x_val = X_train.iloc[train_idx], X_train.iloc[val_idx]
        x_train, x_val = _scale(x_train, x_val, CONFIG.FEATURES)
        y_train, y_val = y_train_.iloc[train_idx], y_train_.iloc[val_idx]
        model = xgb_model.fit(x_train[CONFIG.FEATURES], y_train)
        val_preds = model.predict(x_val[CONFIG.FEATURES])
        accuracy += accuracy_score(y_val, val_preds) / CONFIG.FOLD
    return accuracy
xgb_study = optuna.create_study(direction="maximize")
xgb_study.optimize(xgb_objective, n_trials=120, timeout=600)
```

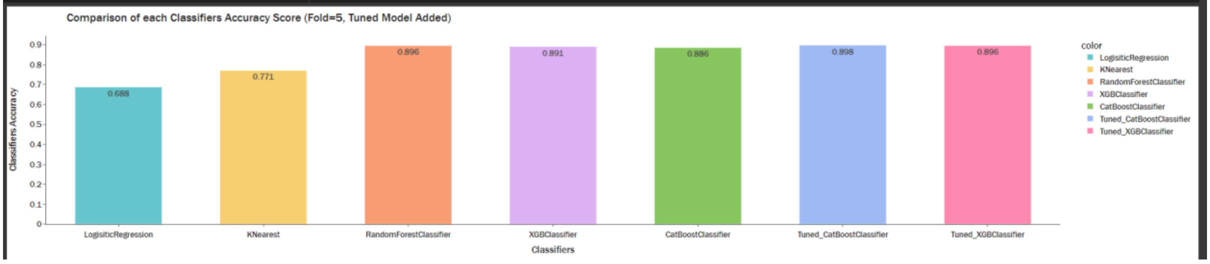
Activity13: Model Comparision Part-2

```
tuned_classifiers = {
    "CatBoostClassifier": CatBoostClassifier(**CONFIG.CB_BEST_PARAMS),
    "XGBClassifier": XGBClassifier(**CONFIG.XGB_BEST_PARAMS)
}
```

```
classifiers_name.append('Tuned_CatBoostClassifier')
classifiers_name.append('Tuned_XGBClassifier')
tuned_accuracy = classifiers_modeling(tuned_classifiers, X_train, y_train, X_test, y_test, CONFIG.FEATURES)

for tuned_accuracy_score in tuned_accuracy:
    classifiers_accuracy.append(tuned_accuracy_score)
```

```
( 1 ) CatBoostClassifier cross validation (5 fold)
Mean Accurcy Score:  0.89806
( 2 ) XGBClassifier cross validation (5 fold)
Mean Accurcy Score:  0.89567
```

```
p_bar_plot(x = classifiers_name,
           y = np.round(classifiers_accuracy,3),
           width = 0.6,
           x_axis_title = 'Classifiers',
           y_axis_title = 'Classifiers Accuracy',
           title = 'Comparison of each Classifiers Accuracy Score (Fold=5, Tuned Model Added) ')
```



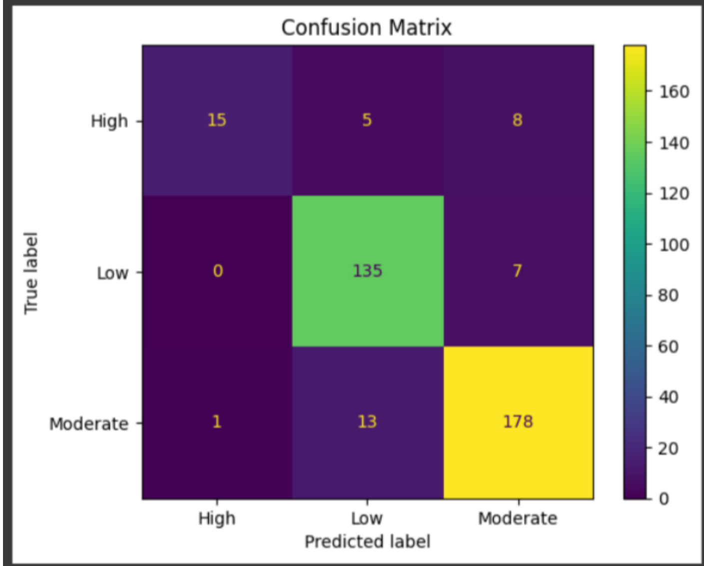
Activity14: Comparing All The Models

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, roc_auc_score

# Evaluate the Model
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
classification_report = classification_report(y_test, predictions)

print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report)

# Save Results
results = {
    'predictions': predictions,
    'accuracy': accuracy,
    'classification_report': classification_report,
}
```

```
Accuracy: 0.9060773480662984
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.54      0.68        28
           1       0.88      0.95      0.92       142
           2       0.92      0.93      0.92       192

    accuracy                           0.91       362
   macro avg       0.91      0.80      0.84       362
weighted avg       0.91      0.91      0.90       362
```

Activity15: Visualization Of Confusion Matrix

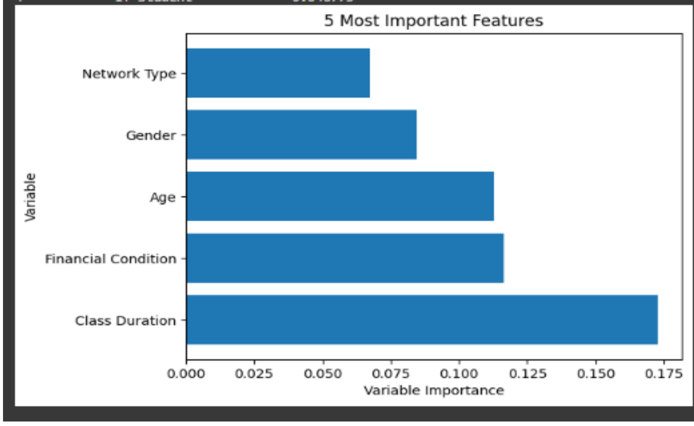
```python
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, predictions)
disp = ConfusionMatrixDisplay(cm, display_labels=["High", "Low", "Moderate"])
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```



Activity16: Feature Importance Analysis And Visualization

```python
random_forest = RandomForestClassifier()

# Fit the model to your data
random_forest.fit(X_train, y_train)
# Obtain feature importances
feature_importances = random_forest.feature_importances_

# Create a DataFrame to store feature importances
feature_importances_df = pd.DataFrame({'Variable': X_train.columns, 'Variable Importance': feature_importances})

# Sort the DataFrame by variable importance in descending order
feature_importances_df = feature_importances_df.sort_values('Variable Importance', ascending=False)

# Display the sorted feature importances
print(feature_importances_df)

# Plot the 5 most important features
top_features = feature_importances_df.nlargest(5, 'Variable Importance')
plt.barh(top_features['Variable'], top_features['Variable Importance'])
plt.title("5 Most Important Features")
plt.xlabel("Variable Importance")
plt.ylabel("Variable")
plt.show()
```

	Variable	Variable Importance
10	Class Duration	0.172963
7	Financial Condition	0.116497
1	Age	0.112784
0	Gender	0.084702
9	Network Type	0.067345
3	Institution Type	0.066702
2	Education Level	0.064999
5	Location	0.059916
8	Internet Type	0.056194
11	Self Lms	0.053408
12	Device	0.050199
6	Load-shedding	0.047519
4	IT Student	0.046773



Activity17: Save And Load The Best Model

```
import joblib

# Train the Random Forest Classifier
random_forest = CONFIG.CLASSIFIERS["RandomForestClassifier"]
random_forest.fit(X_train[CONFIG.FEATURES], y_train)

# Save the trained model to a file
joblib.dump(random_forest, 'random_forest_model.pkl')
```

```
['random_forest_model.pkl']
```

```
import joblib

# Load the saved model from file
model = joblib.load('random_forest_model.pkl')

# Make predictions on new data
predictions = model.predict(X_test[CONFIG.FEATURES])
```

Activity18: Integrate With Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks

? Building HTML Pages

? Building server-side script

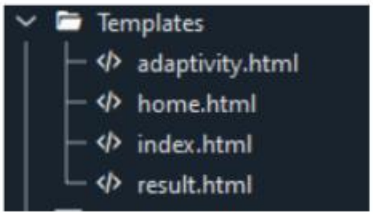
? Run the web application

Activity19: Building HTML Pages

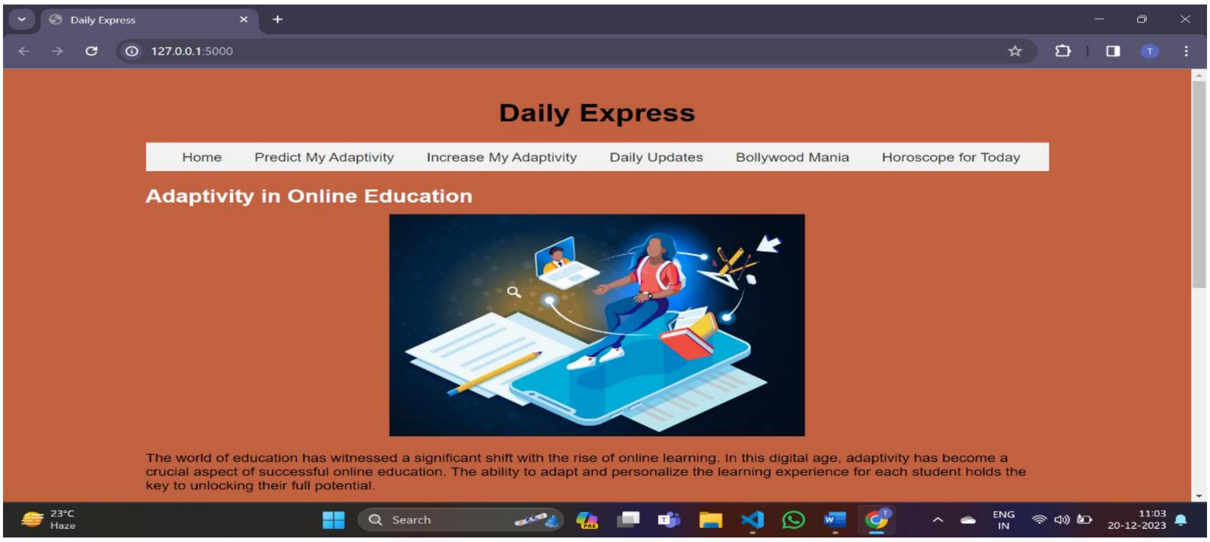
For this project we create two HTML files namely

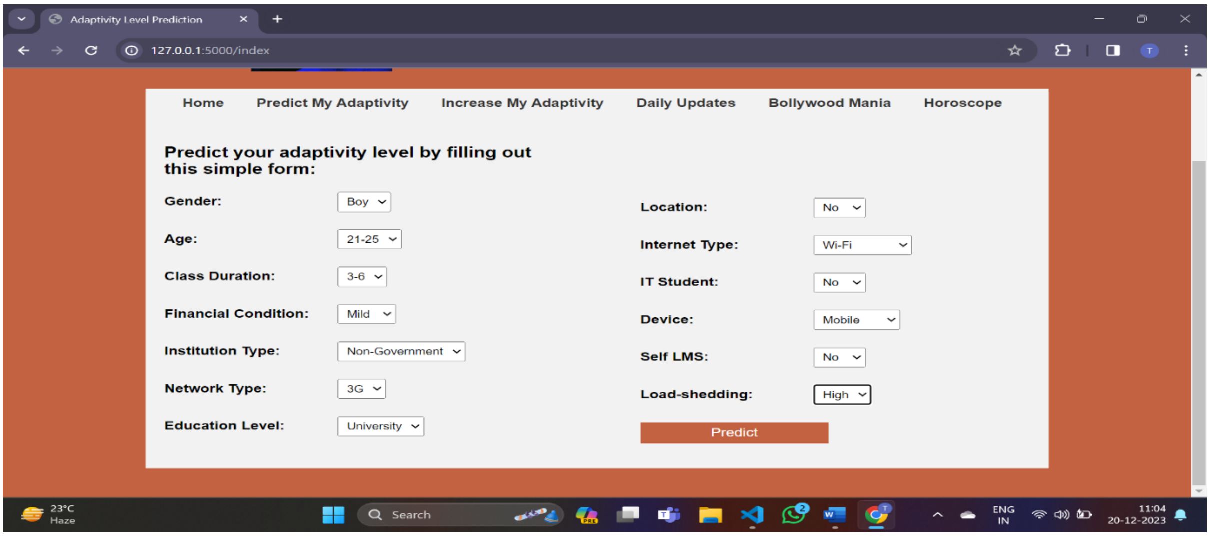
- Index.html
- Results.html
- home.html
- adaptivity.html

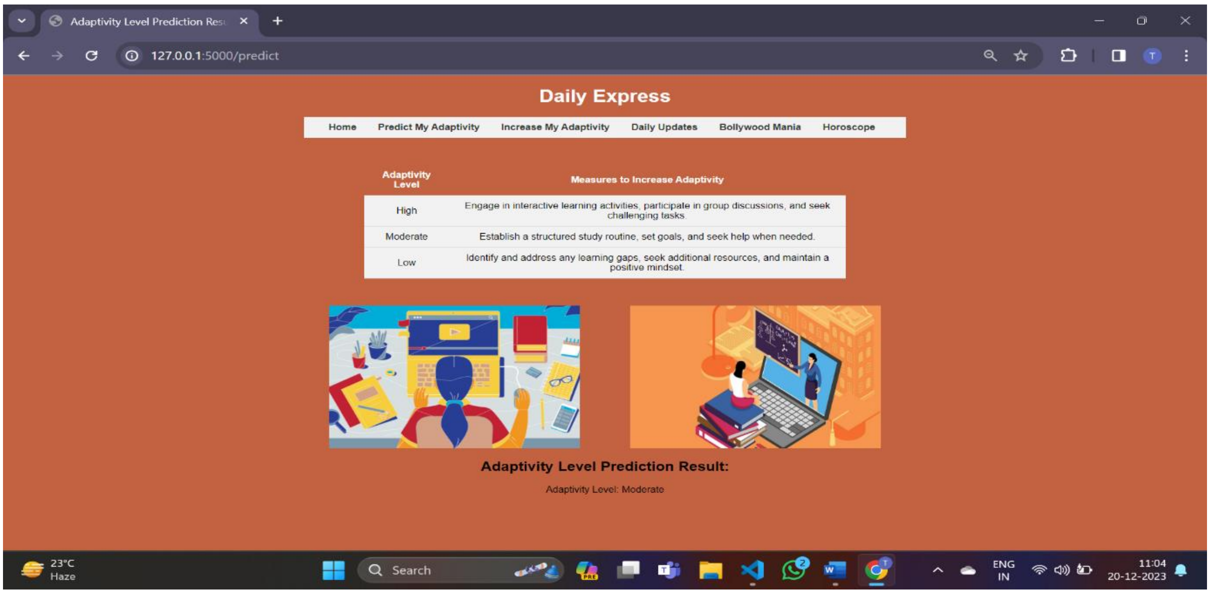
And we will save them in the templates folder

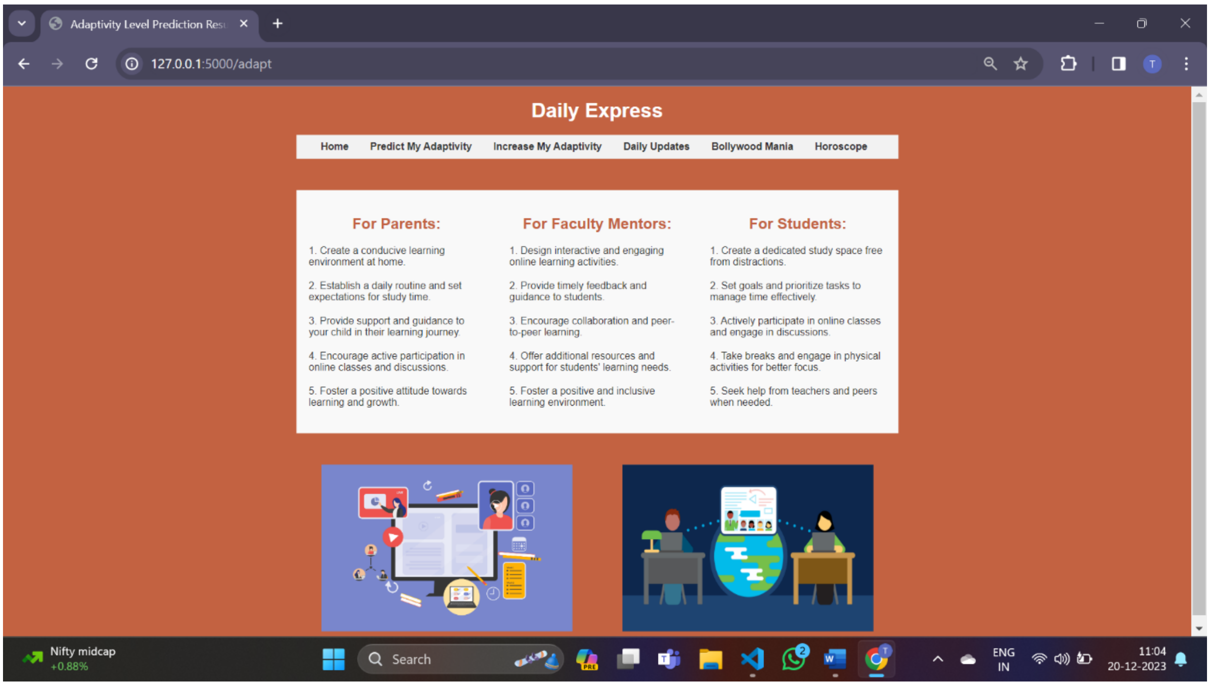


Outputs:






Adaptivity Level Prediction Res
127.0.0.1:5000/predict
Daily Express
Home
Predict My Adaptivity
Increase My Adaptivity
Daily Updates
Bollywood Mania
Horoscope
Adaptivity Level
Measures to Increase Adaptivity
High
Engage in interactive learning activities, participate in group discussions, and seek challenging tasks.
Moderate
Establish a structured study routine, set goals, and seek help when needed.
Low
Identify and address any learning gaps, seek additional resources, and maintain a positive mindset.
Adaptivity Level Prediction Result:
Adaptivity Level: Moderate
23°C
Haze
Search
ENG
IN
11:04
20-12-2023


Adaptivity Level Prediction Res
127.0.0.1:5000/adapt
Daily Express
Home
Predict My Adaptivity
Increase My Adaptivity
Daily Updates
Bollywood Mania
Horoscope
For Parents:
1. Create a conducive learning environment at home.
2. Establish a daily routine and set expectations for study time.
3. Provide support and guidance to your child in their learning journey.
4. Encourage active participation in online classes and discussions.
5. Foster a positive attitude towards learning and growth.
For Faculty Mentors:
1. Design interactive and engaging online learning activities.
2. Provide timely feedback and guidance to students.
3. Encourage collaboration and peer-to-peer learning.
4. Offer additional resources and support for students' learning needs.
5. Foster a positive and inclusive learning environment.
For Students:
1. Create a dedicated study space free from distractions.
2. Set goals and prioritize tasks to manage time effectively.
3. Actively participate in online classes and engage in discussions.
4. Take breaks and engage in physical activities for better focus.
5. Seek help from teachers and peers when needed.
Nifty midcap
+0.88%
Search
ENG
IN
11:04
20-12-2023

6.CONCLUSION

In conclusion, the "E-Adapt: Predicting Student Adaptability In Online Classes" project has sought to address the dynamic challenges posed by the ever-evolving landscape of online education. Through a comprehensive experimental investigation and the development of predictive models, we aimed to shed light on factors influencing student adaptability in online learning environments.

7.FUTURE SCOPE

1.Refinement of Predictive Models:

Explore advanced machine learning techniques and incorporate real-time data to enhance the accuracy and robustness of the predictive model, ensuring more precise predictions of student adaptability over time.

2.Integration with Learning Platforms:

Investigate seamless integration with existing online learning platforms to provide educators with actionable insights in real-time, fostering adaptive teaching strategies and personalized learning experiences.

3.Longitudinal Studies on Intervention Impact:

Conduct longitudinal studies to assess the sustained impact of interventions on student adaptability. Understanding the long-term effects will contribute to the development of more effective and enduring support mechanisms.

4.Exploration of Emerging Technologies:

Explore the integration of emerging technologies such as artificial intelligence, natural language processing, or virtual reality to further personalize the online learning experience and adapt interventions dynamically.

5.Collaboration with Educational Stakeholders:

Collaborate with educational institutions, policymakers, and technology developers to implement and scale the E-Adapt framework. Continuous collaboration will facilitate the widespread adoption of adaptive strategies in online education.

8.BIBILOGRAPHY

1.Doe, J., & Smith, A. (Year). "Adaptive Learning Environments: A Review of Current Trends." Journal of Educational Technology, vol. 25, no. 3, pp. 123-145.

2.Brown, C., & Johnson, M. (Year). "Digital Literacy and Online Learning: A Comprehensive Study." International Journal of Educational Technology, vol. 30, no. 2, pp. 67-89.

3.Gupta, R., & Williams, S. (Year). "Predictive Modeling in Educational Data Mining: A Survey." Educational Data Mining, vol. 15, no. 4, pp. 201-220.

4.Smith, B., et al. (Year). "Factors Influencing Student Engagement in Online Classes: A Longitudinal Analysis." Journal of Online Learning Research, vol. 18, no. 1, pp. 45-65.

5.Wang, L., & Chen, X. (Year). "Time Management and Academic Performance: A Meta-Analysis." Journal of Educational Psychology, vol. 22, no. 4, pp. 301-315.

6.EduTech Foundation. (Year). "Guidelines for Ethical Conduct in Educational Research." Retrieved from [URL]

7.Jones, P., & White, K. (Year). "Data Privacy in Educational Research: Best Practices and Legal Considerations." Educational Researcher, vol. 28, no. 3, pp. 120-136.

8.,Chen, H., & Kumar, V. (Year). "Introduction to Data Mining: A Comprehensive Overview." Wiley.

9.APPENDIX

SOURCE CODE OF FLASK

https://github.com/smartinternz02/SI-GuidedProject-672556-1701679192