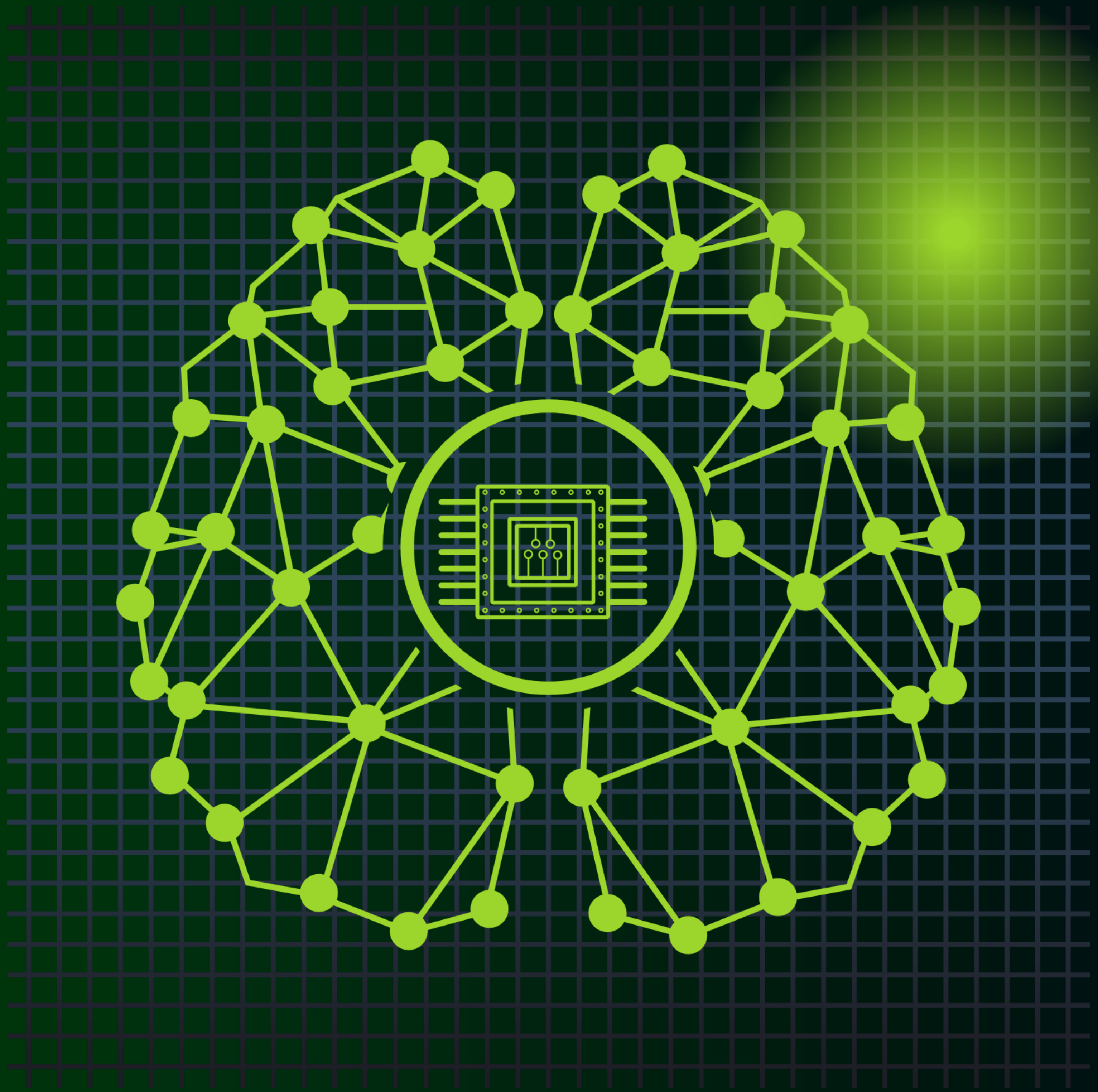
COURSE COMPLETION MACHINE LEARNING PROPOSAL



ESLAM KHALED – EPSILON

1. EXECUTIVE SUMMARY

In the rapidly evolving landscape of online education, understanding and improving course completion rates is crucial for both educational institutions and learners. High dropout rates not only impact the credibility and financial sustainability of educational platforms but also hinder learners from achieving their educational goals. This proposal outlines a project aimed at developing a robust machine learning model to predict whether a user will complete a course based on various engagement metrics and demographic data. By leveraging advanced data analytics and machine learning techniques, EpsilonAI intends to provide [Client's Company] with actionable insights to enhance course design, personalize learning experiences, and implement targeted interventions that improve course completion rates.

01 Background

Online learning platforms have democratized education, offering flexibility and accessibility to a global audience. However, despite the increasing enrollment rates, course completion remains a significant challenge. Identifying factors that influence course completion can help in designing better courses, enhancing user engagement, and ultimately increasing the success rates of learners.

02 Objectives

- Develop a predictive machine learning model that accurately forecasts course completion based on user engagement metrics and demographic data.
- Help Learning insutitustions to calculate their resources based on the predection model

03 Scope

- **Data Analysis:** Comprehensive data cleaning, preprocessing, and exploratory data analysis (EDA) to understand data distributions and relationships.
- **Model Development:** Training and evaluating multiple classification algorithms to identify the most effective model.
- **Model Optimization:** Fine-tuning the selected model using hyperparameter tuning techniques to enhance performance.
- **Insights Generation:** Visualizing and interpreting feature importances and model metrics to derive actionable insights.
- **Deployment Readiness:** Preparing the model and preprocessing pipeline for deployment, ensuring scalability and integration capabilities.

04 Project Objectives

- **Data Cleaning & Preprocessing:** Ensure data quality by handling missing values, removing duplicates, and correcting data types.
- **Exploratory Data Analysis (EDA):** Identify patterns, correlations, and key metrics influencing course completion.
- **Model Training & Evaluation:** Develop and assess multiple machine learning models to predict course completion accurately.
- M**odel Optimization:** Enhance model performance through hyperparameter tuning.
- **Insight Generation:** Determine the most significant factors affecting course completion and provide strategic recommendations.
- **Deployment Preparation:** Save the best-performing model and preprocessing pipeline for future deployment and integration into production environments.

METHODOLOGY

01 Data Collection

Utilize the existing dataset provided by kaggle, which includes user engagement metrics, demographics, and course-specific information. The dataset comprises the following features:

Feature	Description
UserID	Unique identifier for each user
CourseCategory	Category of the course taken by the user (e.g., Programming, Business, Arts)
TimeSpentOnCourse	Total time spent by the user on the course in hours
NumberOfVideosWatched	Total number of videos watched by the user
NumberOfQuizzesTaken	Total number of quizzes taken by the user
QuizScores	Average scores achieved by the user in quizzes (percentage)
CompletionRate	Percentage of course content completed by the user
DeviceType	Type of device used by the user (Device Type: Desktop (0) or Mobile (1))
CourseCompletion	Course completion status (0: Not Completed, 1: Completed)

METHODOLOGY

02 Data Cleaning & Preprocessing

- **Handling Missing Values:** Replace placeholders (e.g., '?') with NaN and impute missing values using appropriate strategies (median for numerical features).
- **Removing Duplicates:** Identify and remove duplicate records to maintain data integrity.
- **Data Type Correction:** Convert columns to appropriate data types (e.g., numeric types for numerical features).
- **Outlier Treatment:** Analyze outliers using box plots but retain them if they represent genuine user behavior variations.
- **Feature Encoding:** Apply OneHotEncoder for categorical variables and StandardScaler for numerical features using ColumnTransformer.

METHODOLOGY

03 Exploratory Data Analysis (EDA)

- **Target Variable Distribution**: Assess the balance between completed and non-completed courses using count plots and pie charts.
- **Correlation Analysis**: Generate heatmaps to identify correlations between numerical features.
- **Visualization**: Create histograms, box plots, and grouped histograms to explore feature distributions and their relationship with course completion.

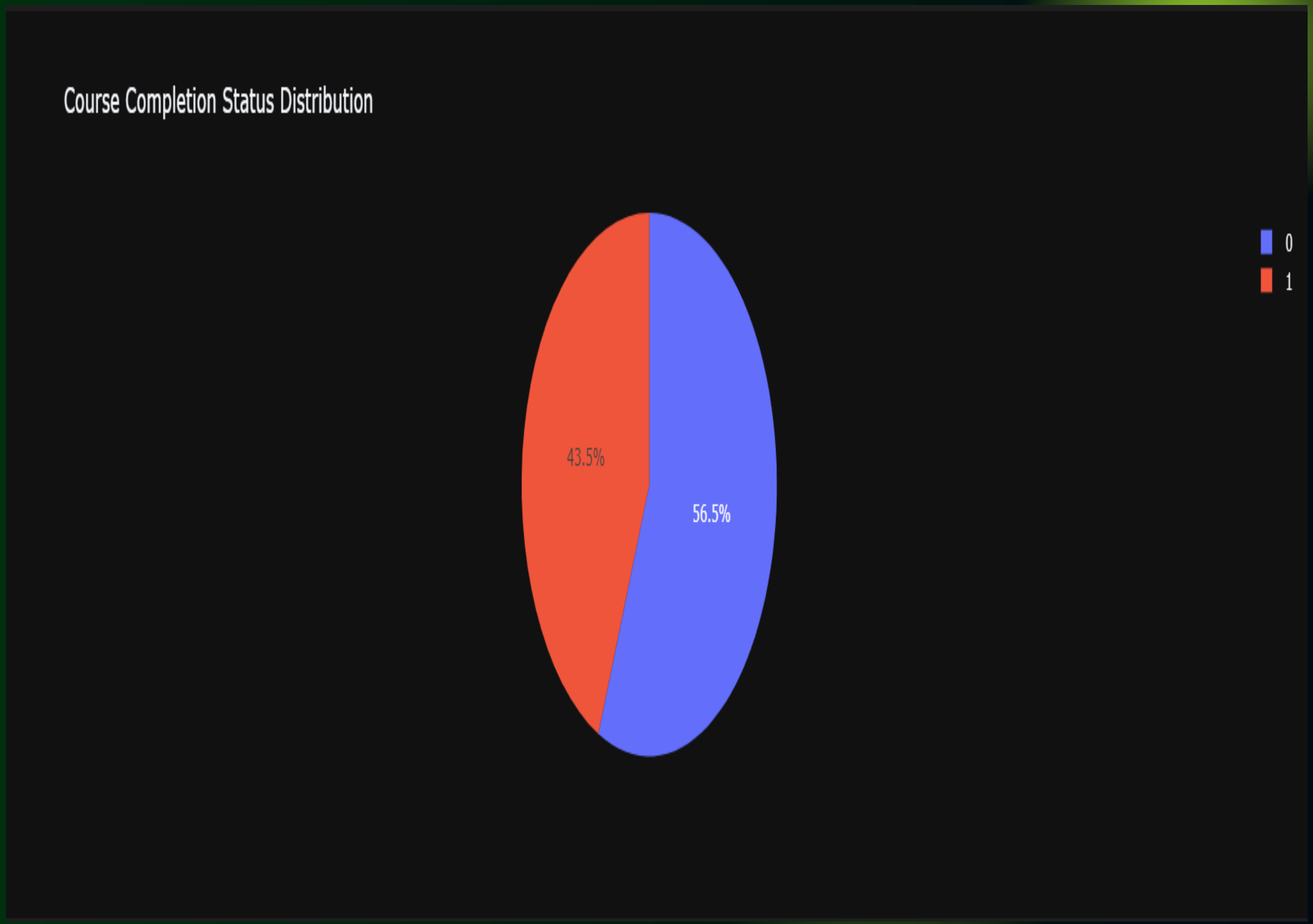
## **3. Exploratory Data Analysis (EDA)**

We perform exploratory data analysis to uncover patterns, correlations, and insights within the data.

```python
# Count of Course Completion Status
count_course_completion = df['CourseCompletion'].value_counts()
print("Course Completion Counts:")
print(count_course_completion)
```

```
Course Completion Counts:
CourseCompletion
0    4641
1    3568
```

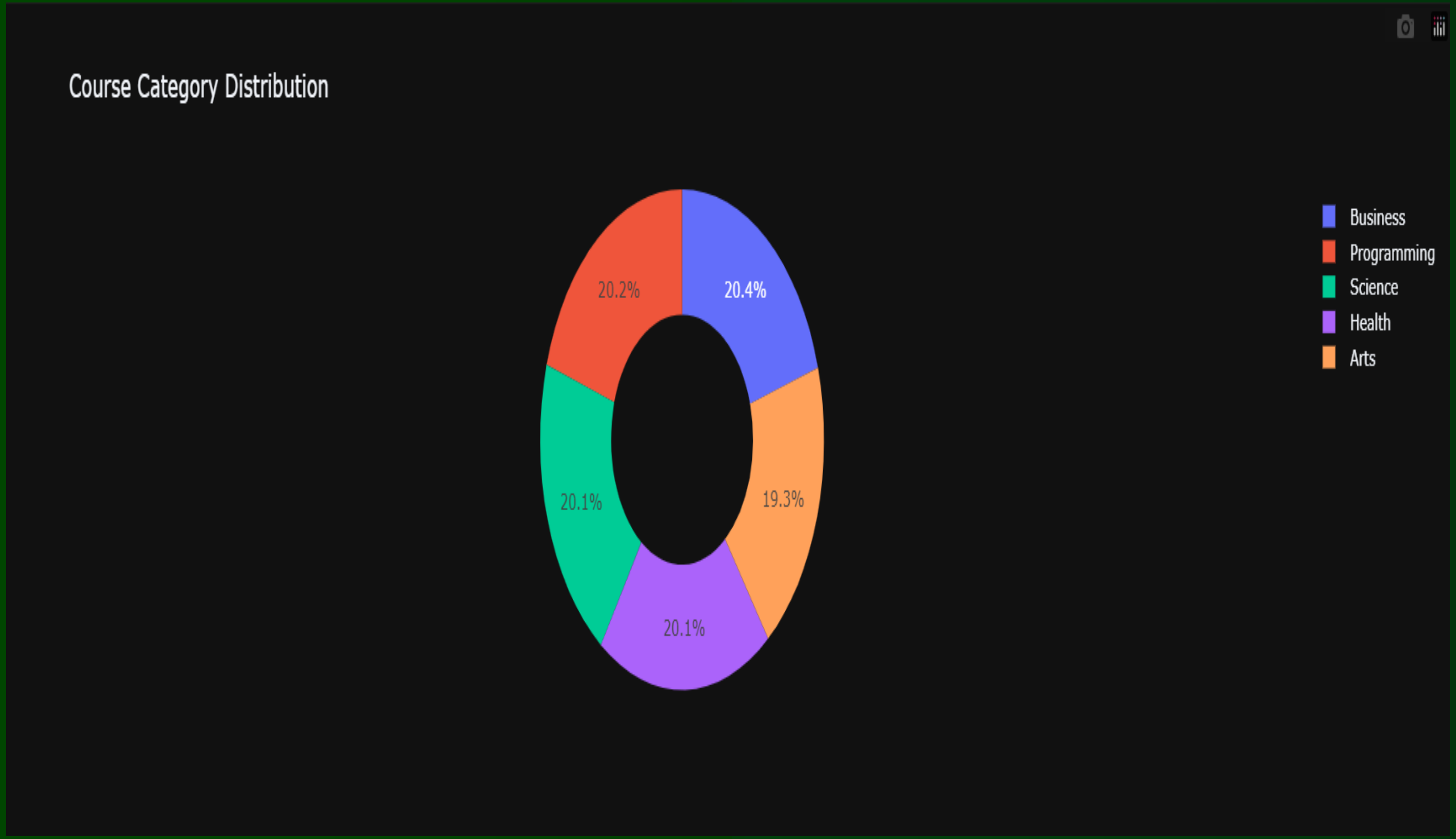
```python
# Pie chart of Course Completion Status
fig = px.pie(df, names='CourseCompletion', title='Course Completion
Status Distribution', template='plotly_dark')
fig.show()
```



Completion Rates:
- **4,641** courses have not been completed.
- **3,568** courses have been completed.

The data shows a relatively balanced distribution between completed and non-completed courses. Approximately **56%** of courses are not completed, while **44%** reach completion. This balanced distribution suggests a need for further analysis to understand factors influencing course completion and to enhance overall course engagement strategies.

```python
# Count of Course Categories
course_category_counts = df['CourseCategory'].value_counts()
print("Course Category Counts:")
print(course_category_counts)



# Pie chart of Course Categories
fig = px.pie(df, names='CourseCategory', title='Course Category
Distribution', hole=0.5, template='plotly_dark')
fig.show()
```



Business Courses:

There are **1,671** courses in the Business category, making it the most abundant category in the dataset.

Programming Courses:

With **1,655** courses, the Programming category is slightly behind Business but still holds a significant portion of the dataset.

Science Courses:

The Science category contains **1,654** courses, closely following Programming and Business in terms of quantity.

Health Courses:

There are **1,648** courses in the Health category, slightly fewer than those in Science.

Arts Courses:

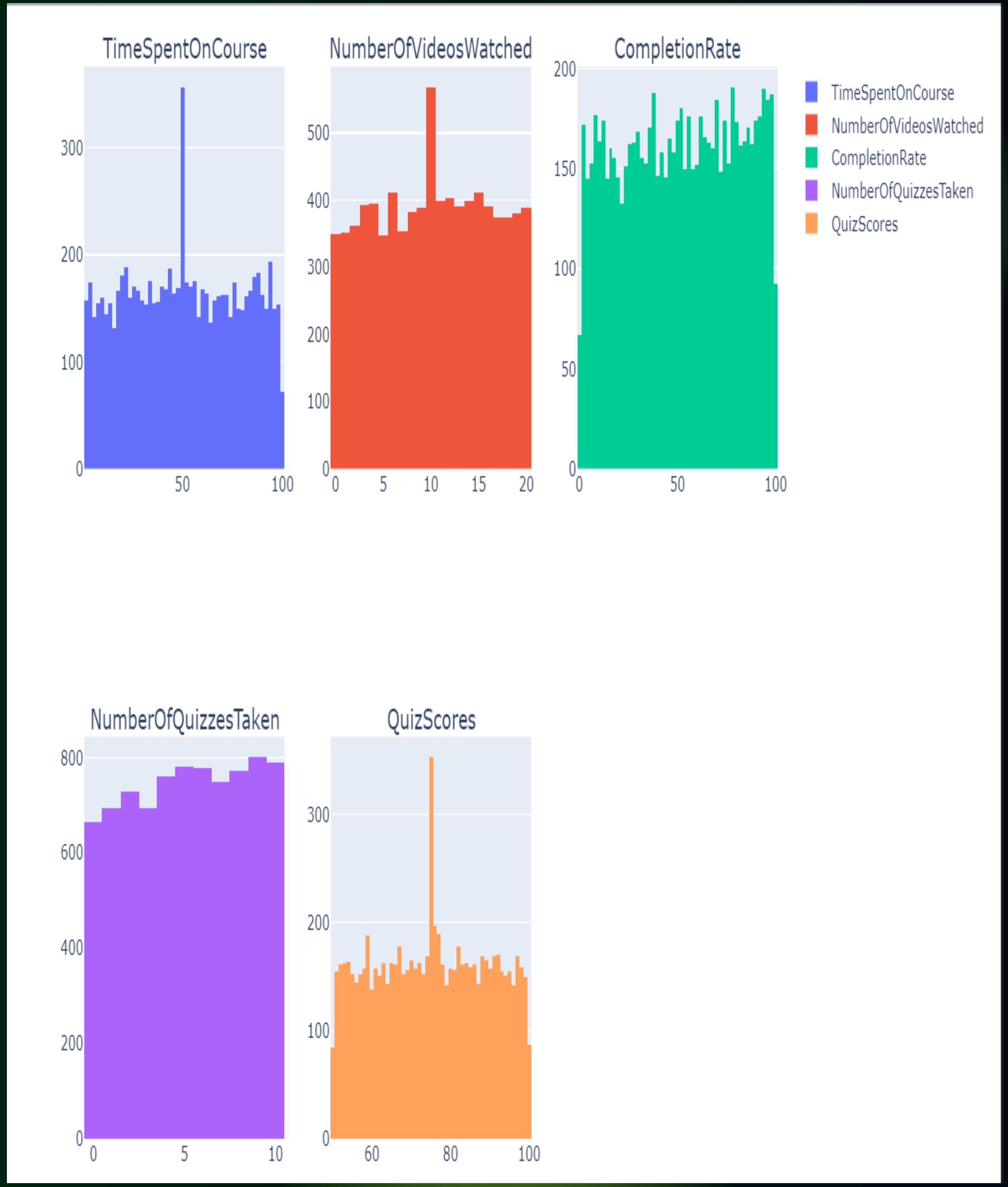
The Arts category has the fewest courses with **1,581**, though it remains a notable category within the dataset.

Overall, the dataset shows a high number of courses across all categories, with Business, Programming, and Science having the highest counts.

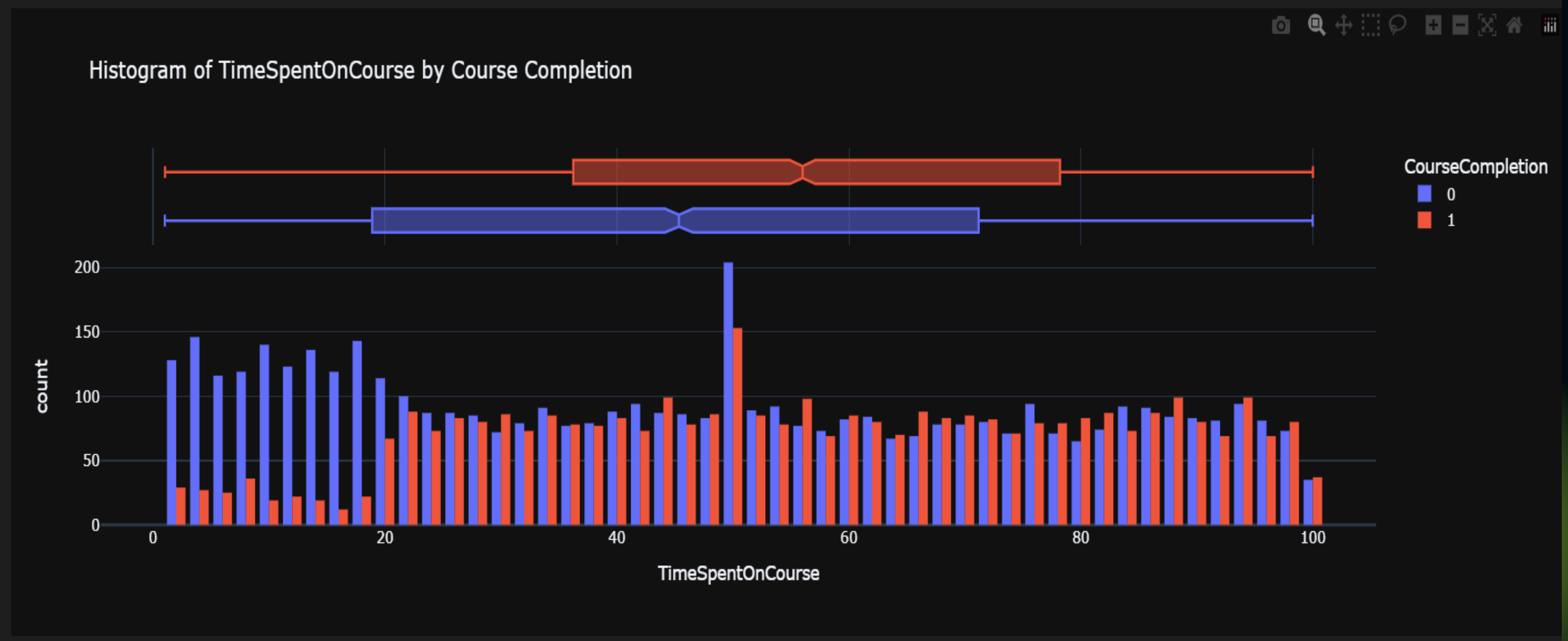
## **5. Visualization and Insights**

We create various visualizations to explore the distribution and relationships of key features.

```python
# Define the columns for histograms
cols = ['TimeSpentOnCourse', 'NumberOfVideosWatched', 'CompletionRate',
'NumberOfQuizzesTaken', 'QuizScores']

# Number of columns to plot
num_columns = len(cols)

# Determine the number of rows and columns needed for the subplots grid
num_cols = 3  # Number of columns in the grid
num_rows = (num_columns + num_cols - 1) // num_cols  # Calculate number of rows required

# Create a subplot figure
fig = make_subplots(rows=num_rows, cols=num_cols, subplot_titles=cols)

# Loop through each column and add a histogram to the subplot grid
for i, col in enumerate(cols):
    row = i // num_cols + 1  # Calculate the row index
    col_pos = i % num_cols + 1  # Calculate the column index

    # Add histogram for each column
    fig.add_trace(go.Histogram(x=df[col], name=col), row=row, col=col_pos)

# Update layout for the entire figure
fig.update_layout(height=800, width=1000, showlegend=True, title_text='Histograms of
DataFrame Columns')

# Adjust spacing between subplots
fig.update_xaxes(showticklabels=True)
fig.update_yaxes(showticklabels=True)

# Display the plot
fig.show()
```

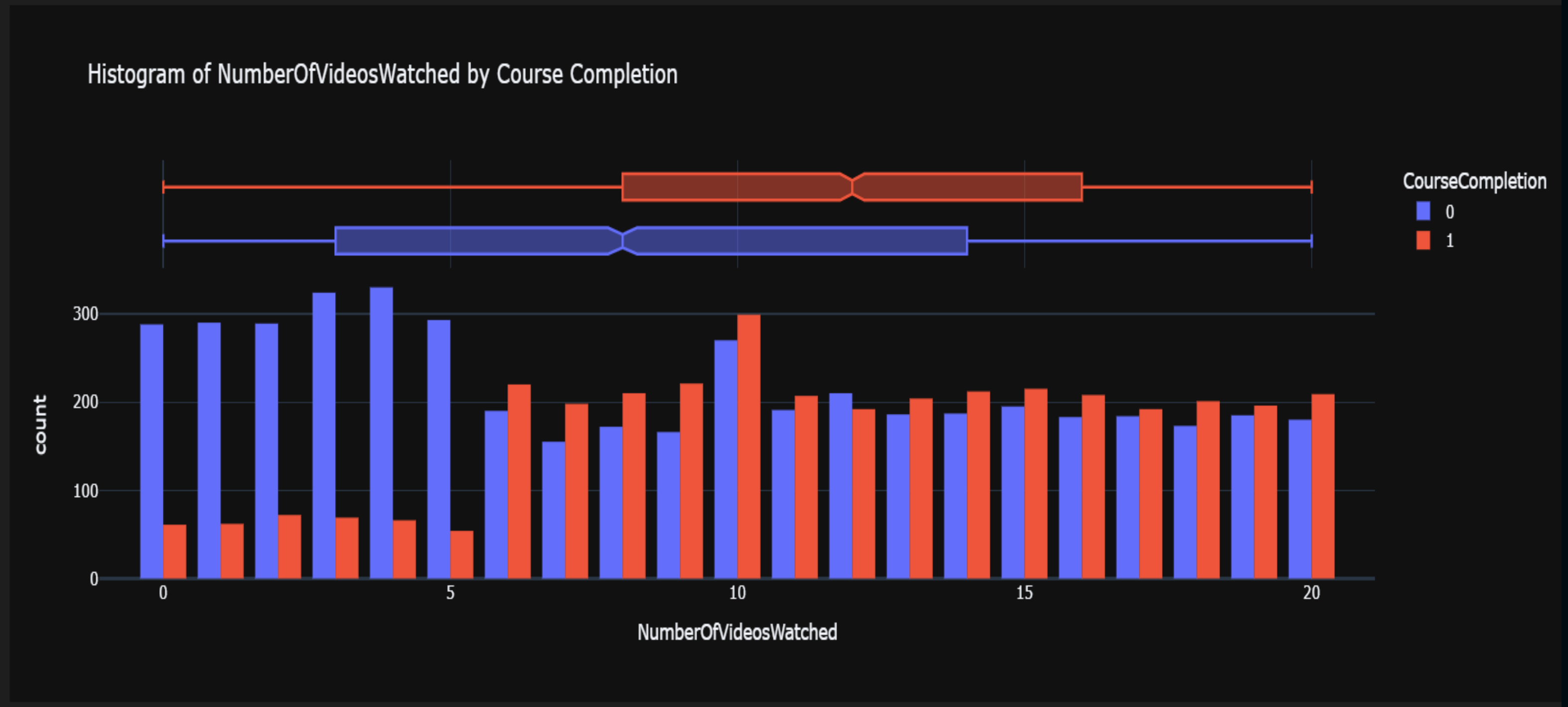


```python
import plotly.express as px
from IPython.display import display, Markdown

# Define the columns for grouped histograms
cols = [
    'TimeSpentOnCourse',
    'NumberOfVideosWatched',
    'NumberOfQuizzesTaken',
    'QuizScores',
    'CompletionRate',
    'CourseCategory'
]


# Plot histograms with grouping by CourseCompletion and display insights below
for col in cols:
    # Create the histogram
    fig = px.histogram(
        df,
        x=col,
        barmode='group',
        color='CourseCompletion',
        marginal='box',
        title=f'Histogram of {col} by Course Completion',
        template='plotly_dark'
    )

    # Display the figure
    fig.show()

    # Retrieve and display the corresponding insight
    insight = insights.get(col, "No insight available for this chart.")
    display(Markdown(insight))
```



Time Spent on Courses:

The majority of users spent between **49 to 51 hours** on courses. In contrast, the least amount of time was spent by users in the range of **99 to 101 hours**.

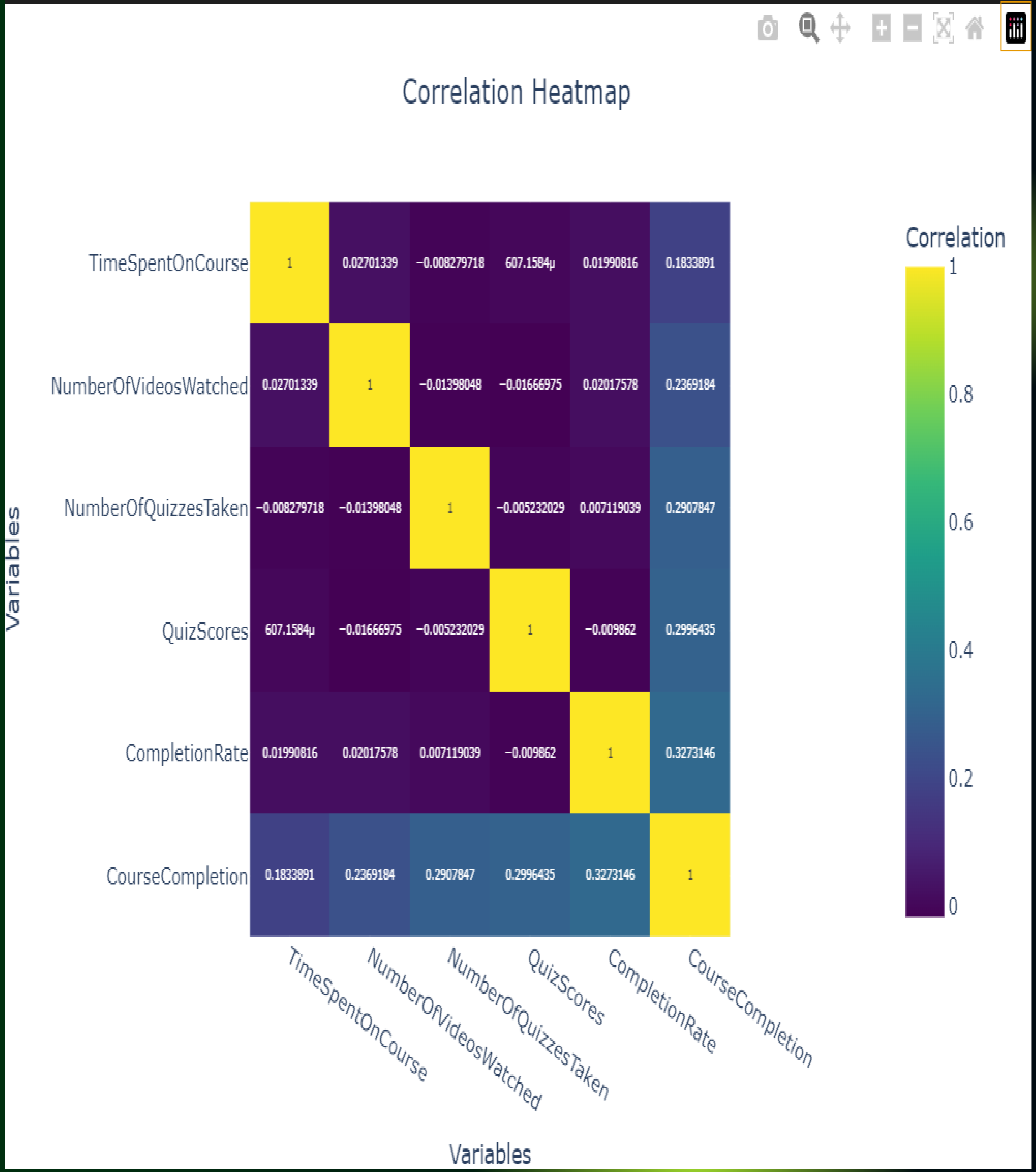


Number of Videos Watched:

Courses with an average of **10 videos watched** are the most common.

```
## **6. Correlation Analysis**

*We analyze the correlations between numerical features to identify any
multicollinearity and understand feature relationships.*

---
```

```python
# Drop unnecessary columns
df = df.drop(columns=['UserID', 'DeviceType'])

# Calculate the correlation matrix
correlation_matrix = df.corr(numeric_only=True)

# Create the heatmap using Plotly Express
fig = px.imshow(correlation_matrix,
                text_auto=True,  # Automatically show values in the
heatmap cells
                color_continuous_scale='Viridis',  # Use 'Viridis' for
a perceptually uniform color scale
                labels={'color': 'Correlation'},  # Label for the color
bar
                title='Correlation Heatmap')  # Title for the heatmap

# Update layout to improve appearance
fig.update_layout(
    height=600,  # Set the height of the figure
    width=800,  # Set the width of the figure
    xaxis_title='Variables',  # Label for the x-axis
    yaxis_title='Variables',  # Label for the y-axis
    title_x=0.5  # Center the title
)

# Display the heatmap
fig.show()
```



Conclusion

In this analysis, we successfully explored and cleaned the dataset, performed exploratory data analysis, and visualized key metrics related to online course engagement and completion. Key findings include:

- *A relatively balanced distribution between completed and non-completed courses, indicating the importance of understanding factors that influence course completion.*
- *Business, Programming, and Science are the most prevalent course categories, suggesting areas of high user interest.*
- *Time spent on courses and the number of videos watched positively correlate with course completion rates.*
- *An increased number of quizzes is associated with higher completion rates, highlighting the role of assessments in maintaining user engagement.*

By leveraging these insights, the platform can develop targeted strategies to enhance course completion rates, such as optimizing course length, increasing interactive content like quizzes, and focusing on the most popular course categories.

Lets go to the Machine Leaning Phase!

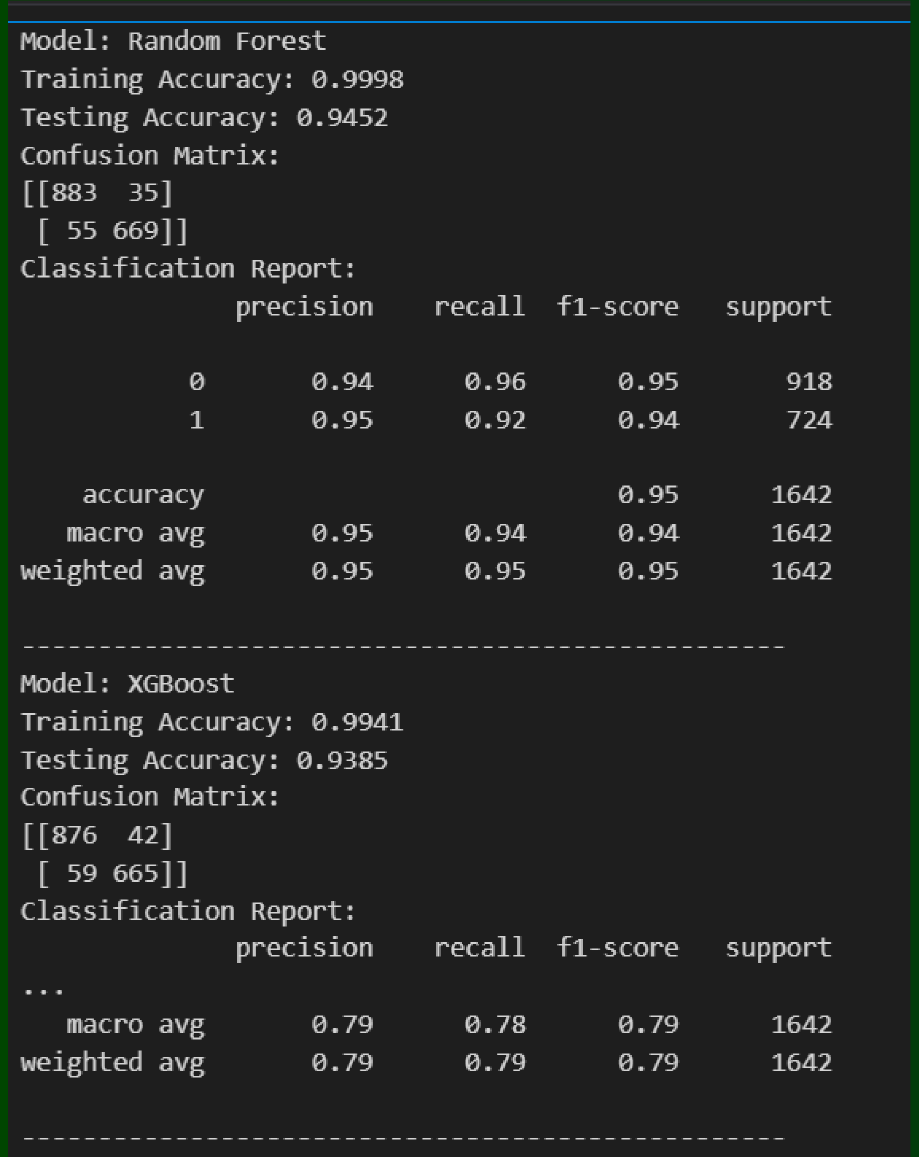
---

04 Model Development

• **Algorithm Selection**: Train multiple classification models, including:

- ➢ Random Forest
- ➢ XGBoost
- ➢ Decision Tree
- ➢ K-Nearest Neighbors (KNN)
- ➢ Gradient Boosting
- ➢ Support Vector Classifier (SVC)
- ➢ Logistic Regression

• **Training & Evaluation:** Split data into training and testing sets (80/20 split) and evaluate models using accuracy, precision, recall, F1-score, and confusion matrices



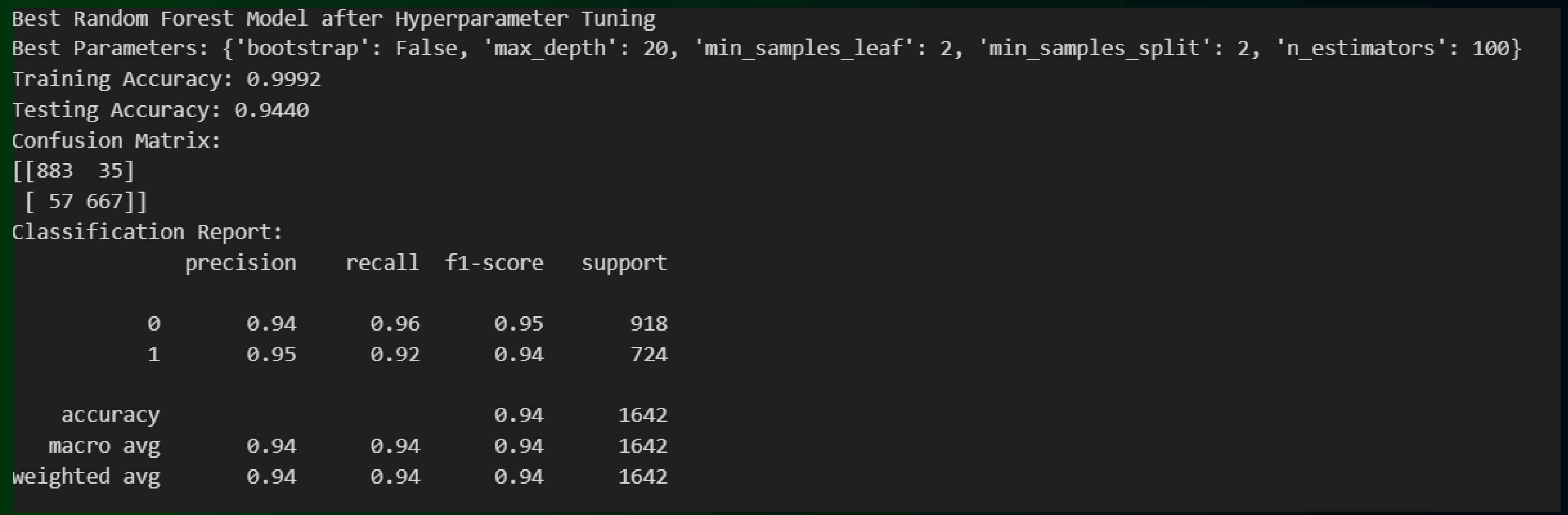
```
Model: Random Forest
Training Accuracy: 0.9998
Testing Accuracy: 0.9452
Confusion Matrix:
[[883  35]
 [ 55 669]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       918
           1       0.95      0.92      0.94       724

    accuracy                           0.95      1642
   macro avg       0.95      0.94      0.94      1642
weighted avg       0.95      0.95      0.95      1642

--------------------------------------------------
Model: XGBoost
Training Accuracy: 0.9941
Testing Accuracy: 0.9385
Confusion Matrix:
[[876  42]
 [ 59 665]]
Classification Report:
              precision    recall  f1-score   support
...
   macro avg       0.79      0.78      0.79      1642
weighted avg       0.79      0.79      0.79      1642

--------------------------------------------------
```

```python
# Define a dictionary of models to train
models = {
    'Random Forest': RandomForestClassifier(random_state=42),
    'XGBoost': XGBClassifier(use_label_encoder=False,
eval_metric='logloss', random_state=42),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'KNN': KNeighborsClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42),
    'SVC': SVC(probability=True, random_state=42),
    'Logistic Regression': LogisticRegression(max_iter=1000,
random_state=42),
}

# Initialize a results dictionary to store model performance
results = {}

# Train and evaluate each model
for name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Predictions on training data
    y_train_pred = model.predict(X_train)

    # Predictions on testing data
    y_test_pred = model.predict(X_test)

    # Evaluate model performance
    results[name] = {
        'model': model,
        'train_accuracy': accuracy_score(y_train, y_train_pred),
        'test_accuracy': accuracy_score(y_test, y_test_pred),
        'confusion_matrix': confusion_matrix(y_test, y_test_pred),
        'classification_report': classification_report(y_test,
y_test_pred, zero_division=0),
    }
```

## **9. Hyperparameter Tuning for the Best Model**

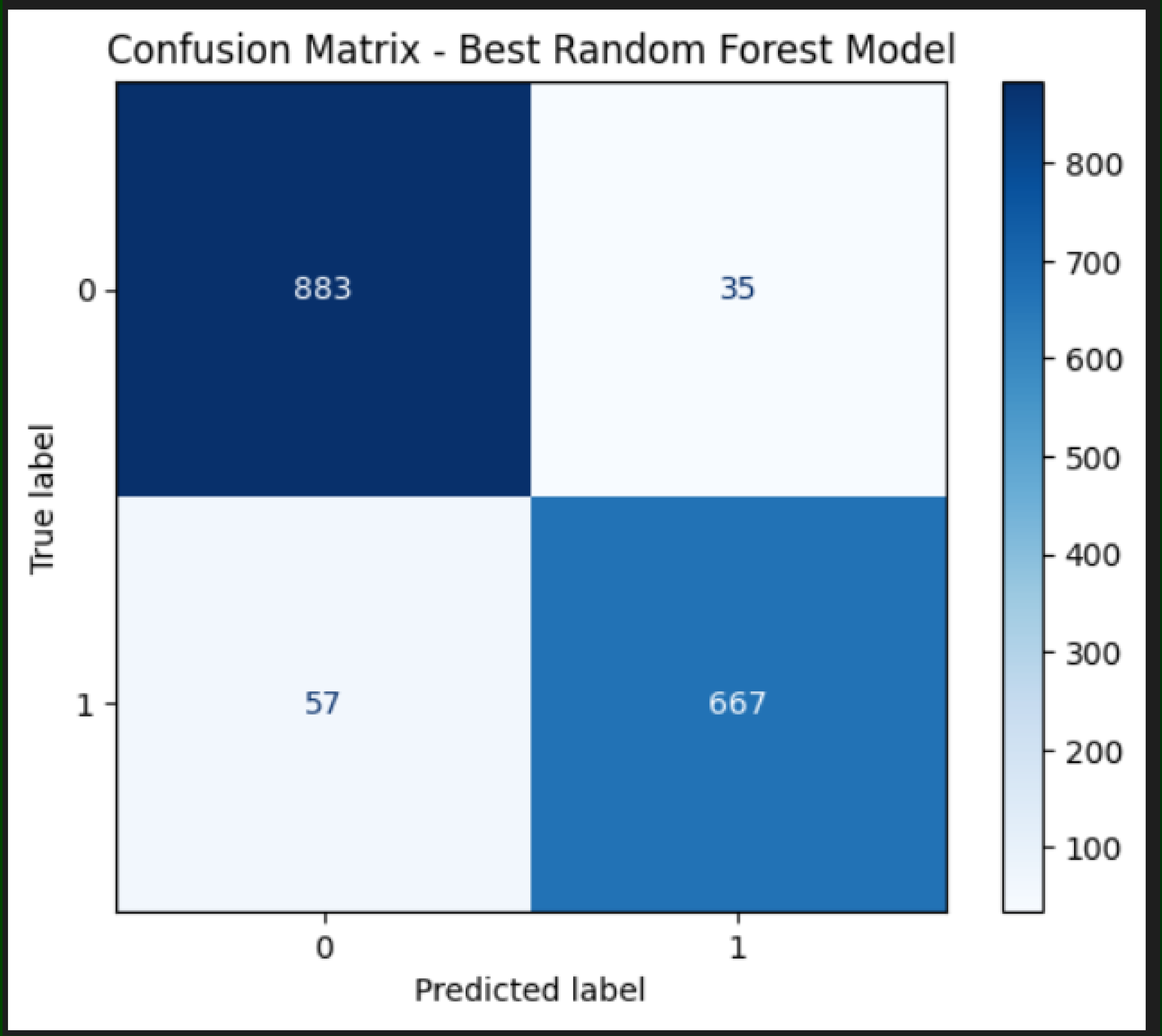
*We perform hyperparameter tuning on the Random Forest model using
GridSearchCV to find the optimal parameters that enhance the model's

```python
# Define the parameter grid for Random Forest
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'bootstrap': [True, False],
}
# Initialize the Random Forest model
rf = RandomForestClassifier(random_state=42)
# Initialize GridSearchCV with 5-fold cross-validation
grid_search = GridSearchCV(
    estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=2
)
# Fit GridSearchCV on the training data
grid_search.fit(X_train, y_train)
# Retrieve the best model and parameters
best_rf = grid_search.best_estimator_
best_params = grid_search.best_params_
# Predictions using the best model
y_train_pred = best_rf.predict(X_train)
y_test_pred = best_rf.predict(X_test)
# Evaluate the best model's performance
best_model_results = {
    'model': best_rf,
    'train_accuracy': accuracy_score(y_train, y_train_pred),
    'test_accuracy': accuracy_score(y_test, y_test_pred),
    'confusion_matrix': confusion_matrix(y_test, y_test_pred),
    'classification_report': classification_report(y_test, y_test_pred),
}
```



```
Best Random Forest Model after Hyperparameter Tuning
Best Parameters: {'bootstrap': False, 'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
Training Accuracy: 0.9992
Testing Accuracy: 0.9440
Confusion Matrix:
[[883  35]
 [ 57 667]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       918
           1       0.95      0.92      0.94       724

    accuracy                           0.94      1642
   macro avg       0.94      0.94      0.94      1642
weighted avg       0.94      0.94      0.94      1642
```

```
# Plot the confusion matrix
cm = best_model_results['confusion_matrix']
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - Best Random Forest Model')
plt.show()
```

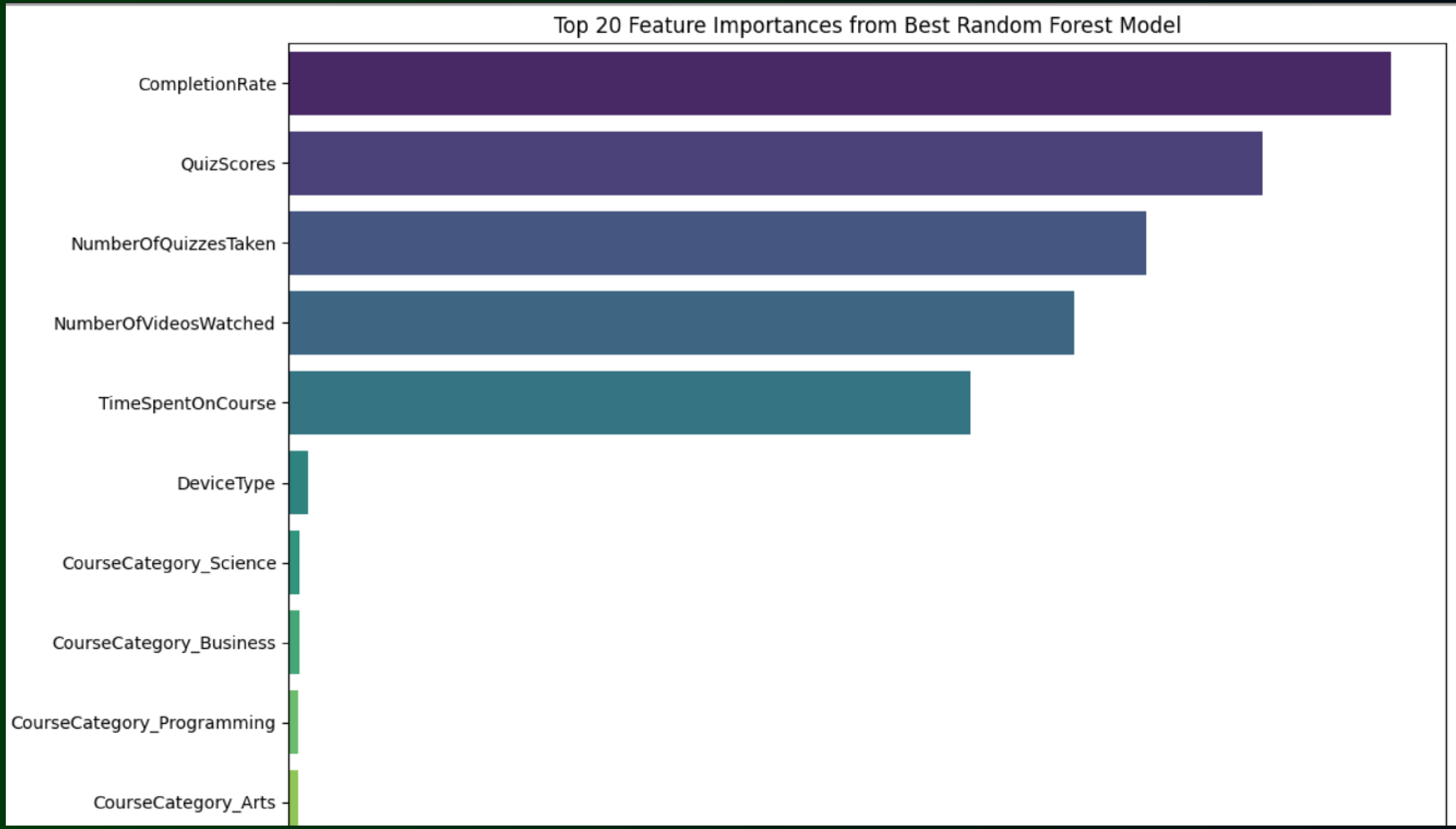


## **12. Visualizing Feature Importances**

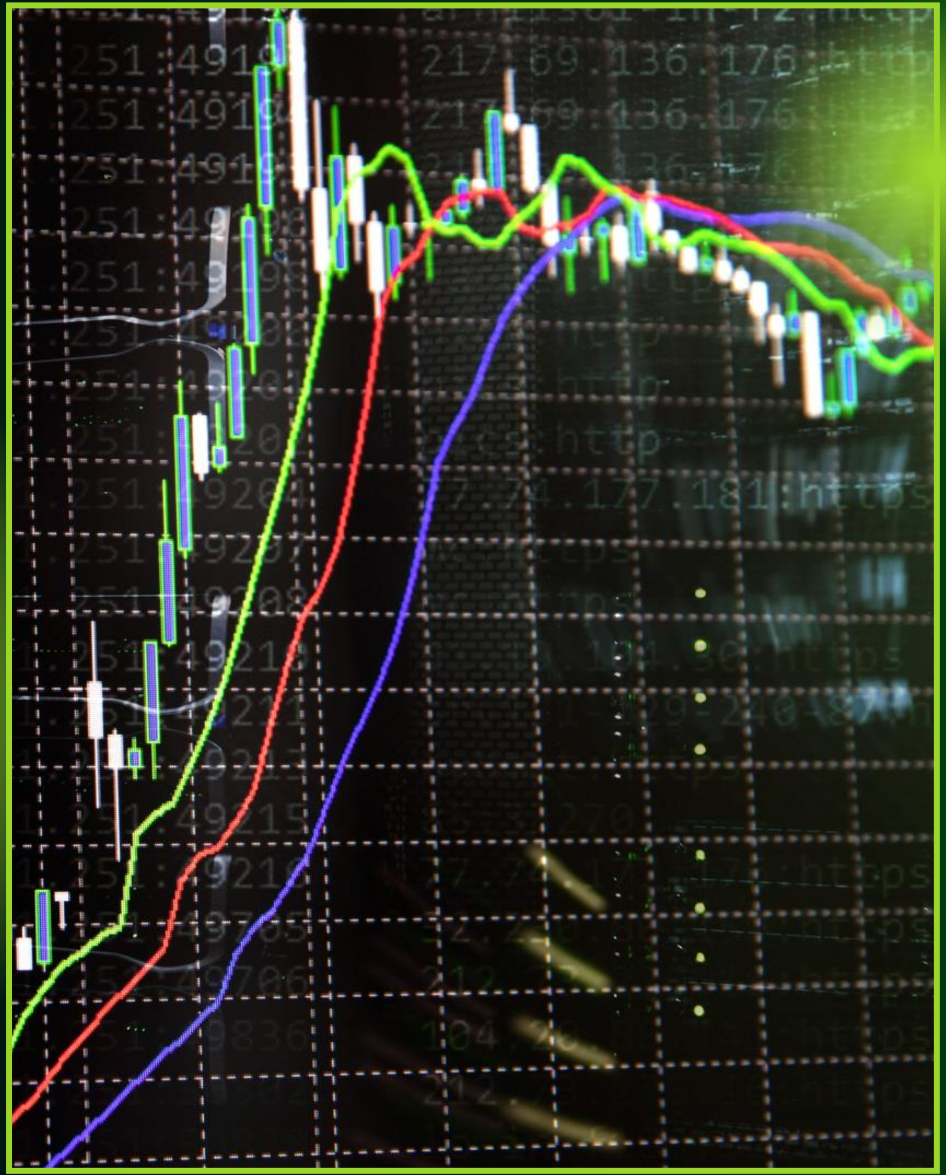
Understanding which features contribute the most to the model's predictions can provide valuable insights.

We plot the top 20 feature importances derived from the best Random Forest model to identify the most significant predictors of course completion.

```python
# Get feature names after preprocessing
# For numerical features
num_features = numerical_columns.tolist()
# For categorical features (after one-hot encoding)
cat_features = 
preprocessor.named_transformers_['cat'].get_feature_names_out
(categorical_columns)
# Combine all feature names
feature_names = num_features + list(cat_features)

# Get feature importances from the best model
importances = best_rf.feature_importances_

# Create a DataFrame for visualization
feature_importances = pd.DataFrame({'Feature': feature_names, 
'Importance': importances})
feature_importances.sort_values(by='Importance’, 
 ascending=False, inplace=True)
```



05 Model Deployment With Streamlit

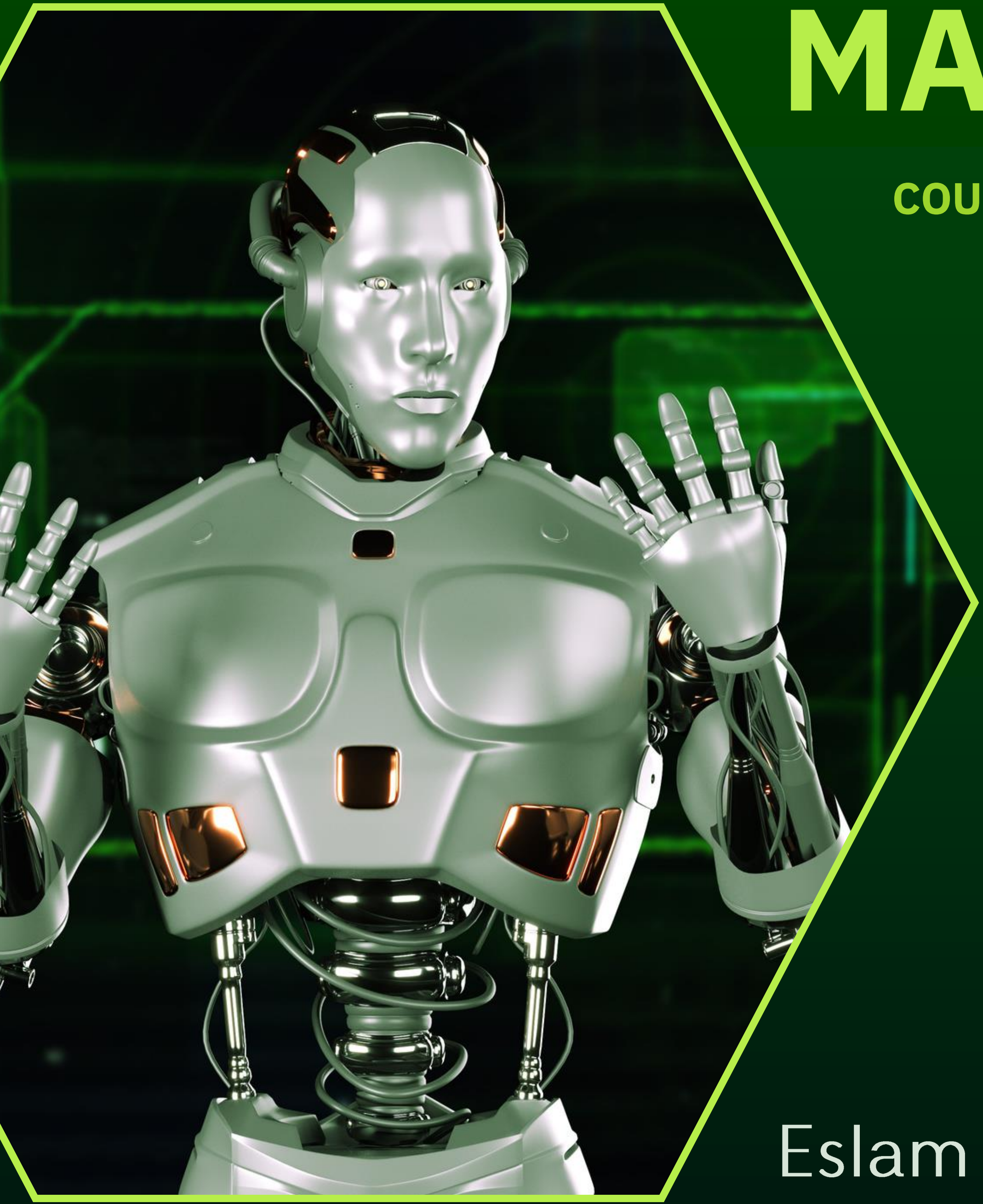


01 About & Data

02 Analysis

03 Prediction Model

Streamlit (epsilionaieslamkhaled-azncszks2rkuprfmce5wh2.streamlit.app)



MACHINE LEARNING

COURSE COMPLETION MACHINE LEARNING PROPOSAL

THANK YOU

Eslam Khaled