National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Umesh Patil
Student ID: X22216481

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mr Umesh E. Patil
Student ID:	X22216481
Programme:	MSc. Data Analytics **Year:** 2023-2024
Module:	Msc Research Project
Lecturer:	Abubakr Siddig
Submission Due Date:	12/08/2024
Project Title:	Enhancing Crowdfunding Success Prediction Using Combinational Approach of Classification and NLP Techniques
Word Count:	763 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Umesh E. Patil
Date:	12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Umesh Patil
Student ID: x22216481

1 Introduction

The configuration manual consists of the step by step guide that have been performed in this research

2 Environment Set Up

2.1 System Specification

- **Processor:** AMD Ryzen 7 5800H with Radeon Graphics, 3201 Mhz, 8 Core(s), 16 Logical Processor(s)
- **Installed RAM:** 16 GB
- **Internal SSD:** 516 GB
- **External SSD:** 1 GB

2.2 Technical Specification

2.2.1 Python

- Version 3.10.12
- In this research Google Colab which is an online python based interface is used. It offers pre-installed libraries, which helps to prevent errors related to version conflicts. Additionally, Google Colab provides powerful GPU support for efficient processing.

3 Data Sources

3.1 Libraries Required

- **Numpy** for numerical operations
- **Pandas** for data manipulation and analysis
- **re** for regular expressions
- **nltk** for natural language processing (e.g., stopwords, lemmatization)
- **matplotlib** for data visualization
- **seaborn** for statistical data visualization
- **scipy** for scientific computing
- **wordcloud** for generating word clouds
- **sklearn.feature_extraction.text.TfidfVectorizer** for text feature extraction
- **sklearn.preprocessing.LabelBinarizer, LabelEncoder** for encoding

- **sklearn** for machine learning algorithms and model evaluation (e.g., ensemble, tree, linear_model, model_selection, metrics)
- **imblearn** for handling imbalanced datasets (e.g., SMOTE)

3.2 Importing Libraries

```
import re
import nltk
import pickle
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
from sklearn import ensemble
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from sklearn.cluster import KMeans
from nltk.stem import PorterStemmer
from imblearn.over_sampling import SMOTE
from scipy.spatial.distance import cdist
from wordcloud import WordCloud, STOPWORDS
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import cross_val_score
```

Figure 1: Importing Libraries

3.3 Data Source and Storage

The data is downloaded from the Kaggle and stored in google drive as shown in Figure 2
Data Link - https://www.kaggle.com/datasets/quentinmcteer/indiegogo-crowdfunding-data

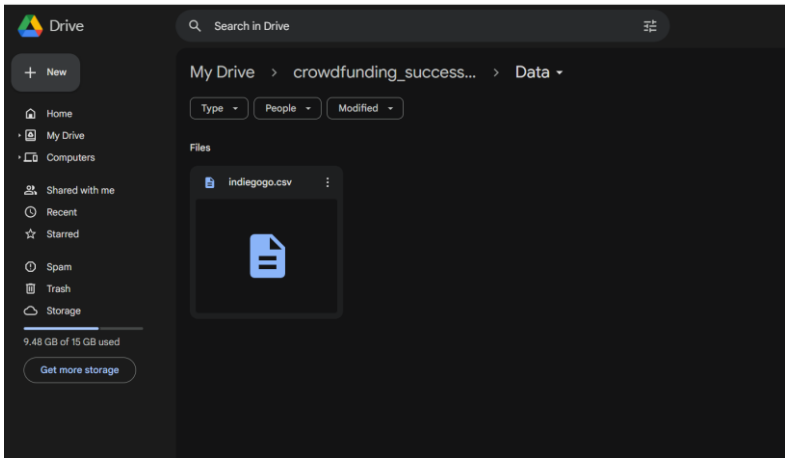


Figure 2: Data Stored in Drive

The data is stored in the google drive and then it is mounted to Google Colab for this research as shown in Figure 3

Dataset Link: /content/drive/MyDrive/crowdfunding_success_prediction/Data/indiegogo.csv

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Figure 3: Mounting Drive

As this process is completed using the pandas library the CSV data file has been read as shown in Figure 4

```
[ ] data = pd.read_csv('/content/drive/MyDrive/crowdfunding_success_prediction/Data/indiegogo.csv')
```

```
[ ] data.shape
```

```
(20631, 74)
```

Figure 4: Reading data using pandas library

4 EDA

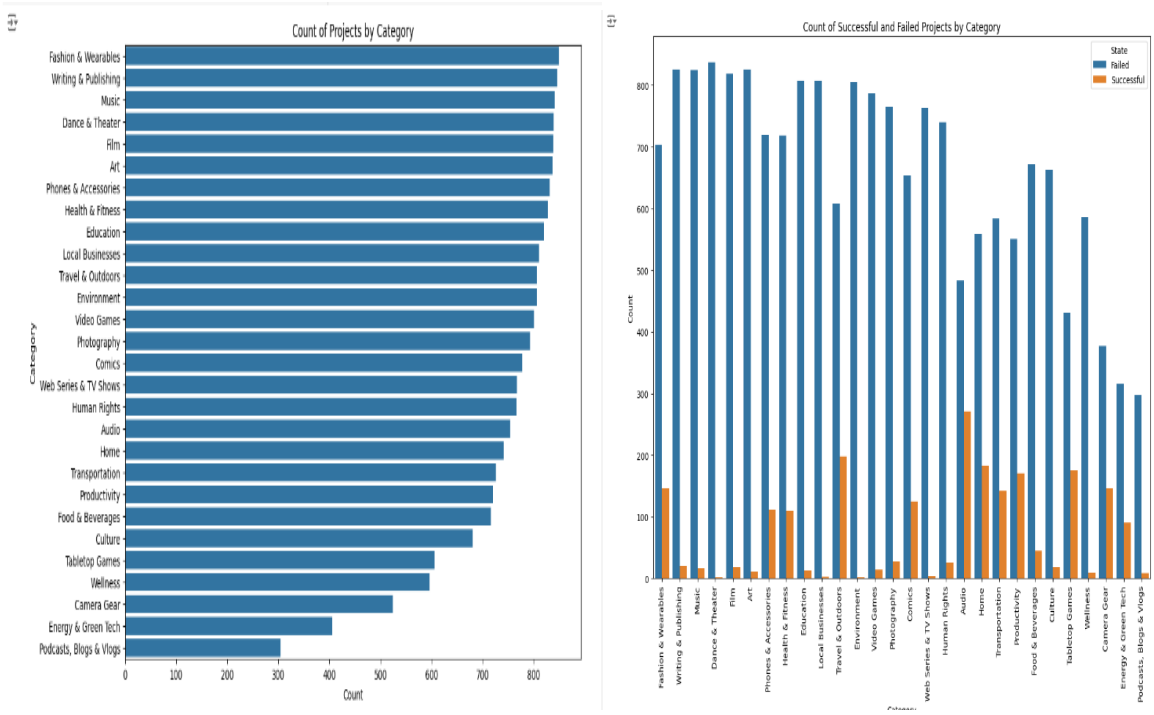


Figure 5: Count of data category-wise and Count of Failed and successful project category wise

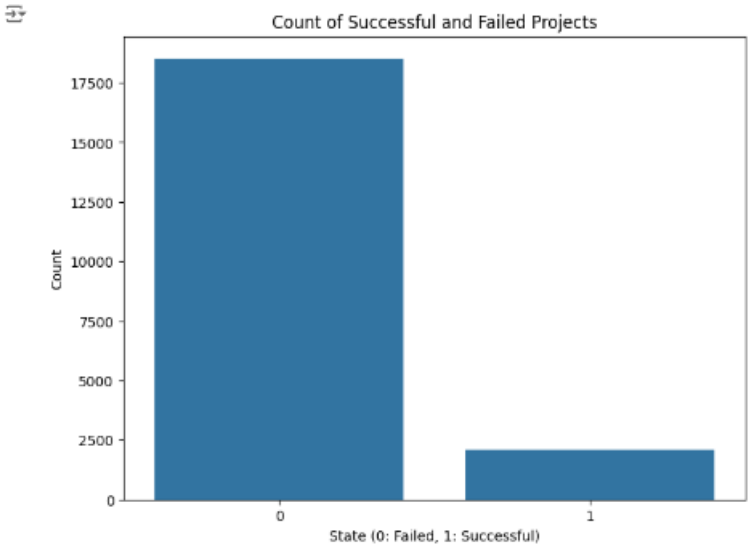


Figure 6: Class Imbalance

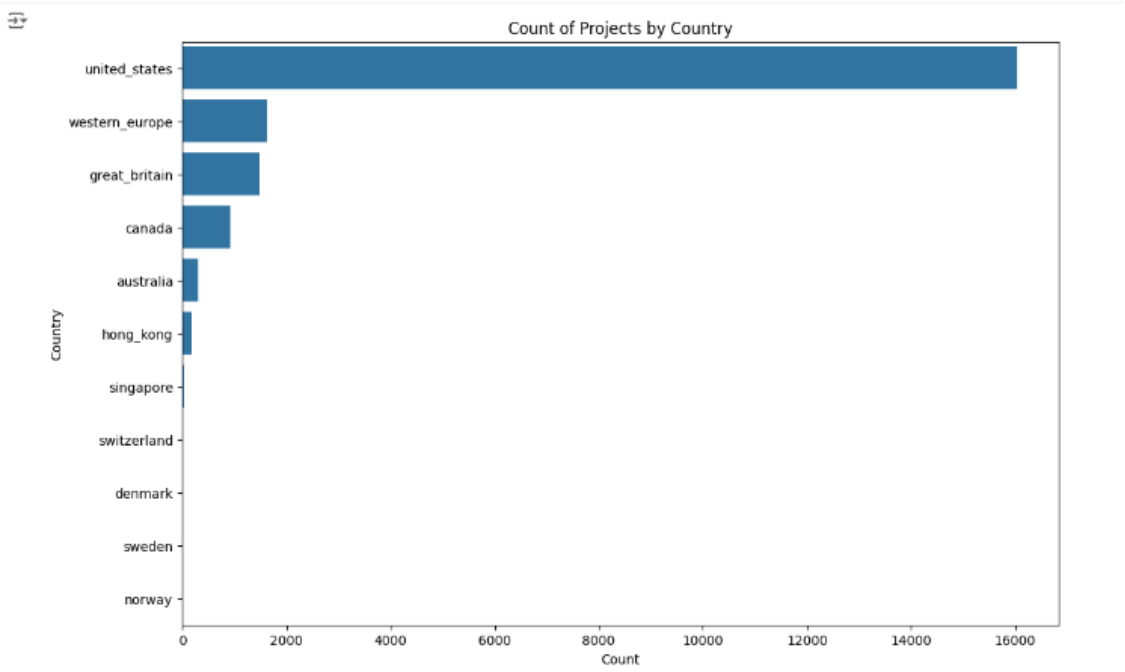


Figure 7: Count of Projects by Country

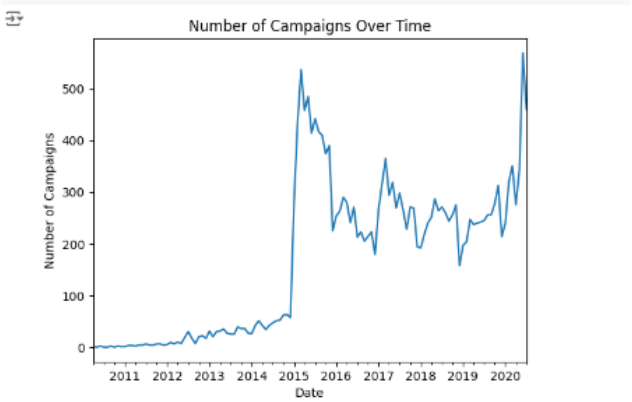


Figure 8: Number of campaigns Over Time

5 TF-IDF Vectorizer

As Shown in Figure The code is used to combine the two columns Title and Tagline and applied TF- IDF Vectorization to carry out textual features.

```
[ ]  from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.decomposition import TruncatedSVD
     # Combine title and tagline for TF-IDF
     new_data['combined_text'] = new_data['title'] + " " + new_data['tagline']

     # Apply TF-IDF
     tfidf = TfidfVectorizer(max_features=1000, min_df=2)
     tfidf_matrix = tfidf.fit_transform(new_data['combined_text'])

     # Reduce dimensionality using Truncated SVD
     svd = TruncatedSVD(n_components=100)  # Reducing to 100 components
     tfidf_reduced = svd.fit_transform(tfidf_matrix)

     # Convert to DataFrame
     tfidf_df = pd.DataFrame(tfidf_reduced)
```

Figure 9: TF-IDF Vectorization

6 Correlation and Normalization

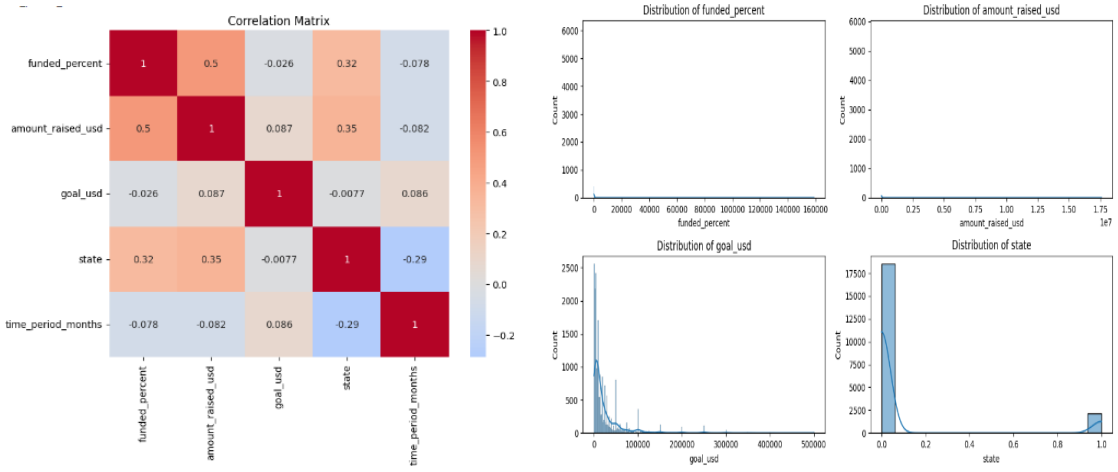


Figure 10: Correlation And data Distribution before Normalization

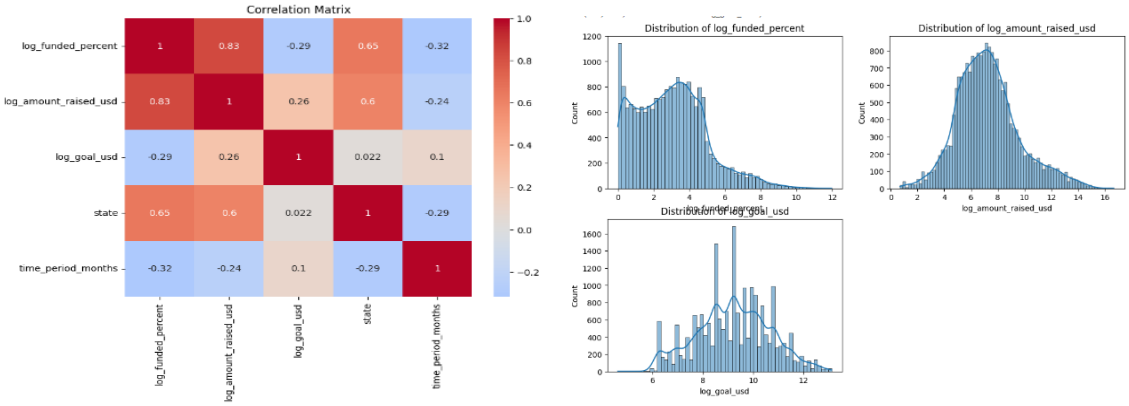


Figure 11: Correlation And data Distribution before Normalization

7 Feature Selection Based on Correlation Matrix and Anova Test

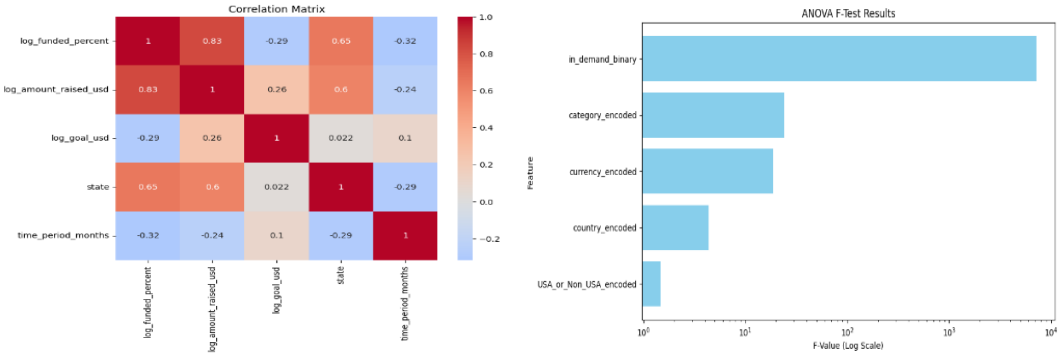


Figure 12: Correlation matrix and ANOVA test Results

8 SMOTE For Imbalance

Figure 13: SMOTE for handling imbalance class

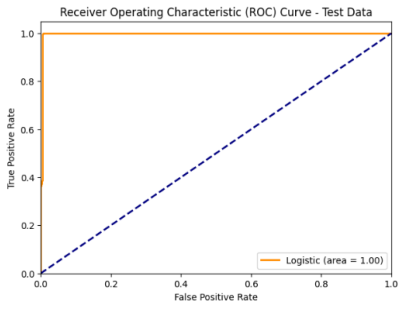
As shown in Figure 13 before model building the imbalance class is handled using SMOTE technique

9 Model Building and Evaluation

9.1 Experiment 1: Model Training and Testing on SMOTE data

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_curve, auc
# Split the SMOTE dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

# Initialize and train the model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
# Predict and evaluate the model on the training set
y_train_pred = model.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
train_precision = precision_score(y_train, y_train_pred)
train_recall = recall_score(y_train, y_train_pred)
train_f1 = f1_score(y_train, y_train_pred)
print(f"Logistic Regression (Training) - Accuracy: {train_accuracy}")
print(f"Logistic Regression (Training) - Precision: {train_precision}")
print(f"Logistic Regression (Training) - Recall: {train_recall}")
print(f"Logistic Regression (Training) - F1 Score: {train_f1}")

# Predict and evaluate the model on the test set
y_test_pred = model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred)
test_recall = recall_score(y_test, y_test_pred)
test_f1 = f1_score(y_test, y_test_pred)

print(f"Logistic Regression (Test) - Accuracy: {test_accuracy}")
print(f"Logistic Regression (Test) - Precision: {test_precision}")
print(f"Logistic Regression (Test) - Recall: {test_recall}")
print(f"Logistic Regression (Test) - F1 Score: {test_f1}")

# Compute ROC curve and ROC area for the test set
y_test_prob = model.predict_proba(X_test)[:, 1]
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_prob)
roc_auc_test = auc(fpr_test, tpr_test)

print(f"Logistic Regression (Test) - AUC: {roc_auc_test}")

# Plot ROC curve for the test set
plt.figure(figsize=(7, 5))
plt.plot(fpr_test, tpr_test, color='darkorange', lw=2, label=f'Logistic (area = {roc_auc_test:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve - Test Data')
plt.legend(loc="lower right")
plt.show()
```

```
Logistic Regression (Training) - Accuracy: 0.9953692749704242
Logistic Regression (Training) - Precision: 0.9908355073917988
Logistic Regression (Training) - Recall: 1.0
Logistic Regression (Training) - F1 Score: 0.9953966600584657
Logistic Regression (Test) - Accuracy: 0.9925645531972421
Logistic Regression (Test) - Precision: 0.985270487412962
Logistic Regression (Test) - Recall: 1.0
Logistic Regression (Test) - F1 Score: 0.9925806016457575
Logistic Regression (Test) - AUC: 0.9971216188415678
```




```
[ ] from sklearn.model_selection import cross_val_score

    # Perform 5-fold cross-validation
    cv_scores = cross_val_score(model, X_smote, y_smote, cv=5, scoring='accuracy')
    print(f"Cross-Validation Accuracy Scores: {cv_scores}")
    print(f"Mean Cross-Validation Accuracy: {cv_scores.mean()}")
```

```
Cross-Validation Accuracy Scores: [0.99878329 0.99891848 0.99634938 0.99067063 0.98053002]
Mean Cross-Validation Accuracy: 0.9930503596059765
```

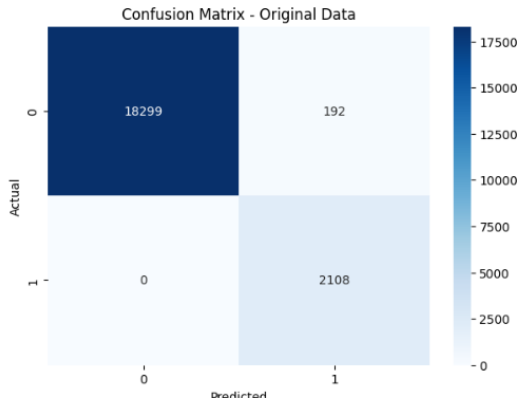
Figure 13: Model Building and testing on SMOTE Data

As shown in Figure 13 the model is trained and tested on SMOTE data. Also using 5 cross validation, and confusion metrics model is evaluated on both test and trained data. Similarly other 5 models are trained and Tested on SMOTE data

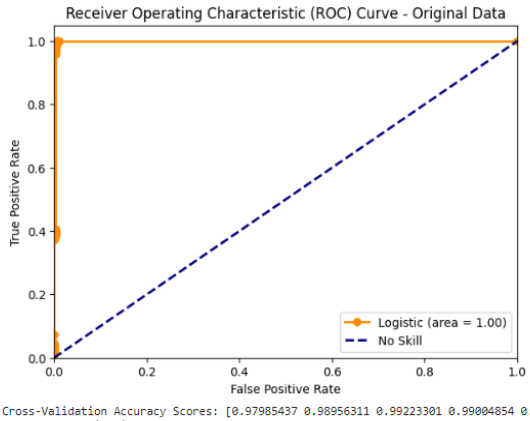
9.2 Experiment 2 : Testing Model on Original Data

```
# Extract features and target from merged_data
X_original = merged_data.drop('state', axis=1).apply(pd.to_numeric, errors='coerce')
y_original = merged_data['state']
# Convert feature names of X_original to strings (if not already done)
X_original.columns = X_original.columns.astype(str)
# Make predictions on the original data
y_original_pred = model.predict(X_original)
# Evaluate the model on the original data
original_accuracy = accuracy_score(y_original, y_original_pred)
original_precision = precision_score(y_original, y_original_pred)
original_recall = recall_score(y_original, y_original_pred)
original_f1 = f1_score(y_original, y_original_pred)

print(f"Logistic Regression (Original Data) - Accuracy: {original_accuracy}")
print(f"Logistic Regression (Original Data) - Precision: {original_precision}")
print(f"Logistic Regression (Original Data) - Recall: {original_recall}")
print(f"Logistic Regression (Original Data) - F1 Score: {original_f1}")

# Confusion matrix for the original data
conf_matrix_original = confusion_matrix(y_original, y_original_pred)

# Plot confusion matrix for the original data
plt.figure(figsize=(7, 5))
sns.heatmap(conf_matrix_original, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix - Original Data')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
# Compute ROC curve and ROC area for the original data
y_original_prob = model.predict_proba(X_original)[:, 1]
fpr, tpr, _ = roc_curve(y_original, y_original_prob)
roc_auc = auc(fpr, tpr)
print(f"Logistic Regression (Original Data) - AUC: {roc_auc}")
# Plot ROC curve
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, marker='o', color='darkorange', lw=2, label=f'Logistic (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--', label='No Skill')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve - Original Data')
plt.legend(loc="lower right")
plt.show()
# Perform 5-fold cross-validation
cv_scores = cross_val_score(model, X_original, y_original, cv=5, scoring='accuracy')
print(f"Cross-Validation Accuracy Scores: {cv_scores}")
print(f"Mean Cross-Validation Accuracy: {cv_scores.mean()}")
```

```
Logistic Regression (Original Data) - Accuracy: 0.9906791591824846
Logistic Regression (Original Data) - Precision: 0.9165217391304348
Logistic Regression (Original Data) - Recall: 1.0
Logistic Regression (Original Data) - F1 Score: 0.956442831215971
```




```
Logistic Regression (Original Data) - AUC: 0.9978810913396814
```




```
Cross-Validation Accuracy Scores: [0.97985437 0.98956311 0.99223301 0.99004854 0.98130614]
Mean Cross-Validation Accuracy: 0.9866010342787508
```

Figure 14: Model Testing on Original Dataset

As shown in Figure 14 model is tested on Original imbalanced data.Similarly other models testings are carried out