National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Venkata Naveen Meka
Student ID: 22206400

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Venkata Naveen Meka
Student ID:	22206400
Programme:	MSc. Data Analytics **Year:** 1
Module:	Research Project
Lecturer:	Jorge Basilio
Submission Due Date:	12/08/2024
Project Title:	Advanced Predictive Modelling of E-Commerce Customer Behaviour: Integrating Machine Learning and Deep Learning Techniques
Word Count:	965 **Page Count:** 11

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:	M V Naveen
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

Venkata Naveen Meka
22206400

1. Introduction:

This manual provides a detailed guide for setting up and running the machine learning models related to cart abandonment and purchase prediction using various classifiers such as Random Forest, LSTM-RF, XGBoost, and Bi-LSTM. It covers the environment setup, data preparation, model training, and testing processes. This manual aims to assist users in reproducing the experiments and understanding the workflow behind the predictions made in this project.

2. Environment Used:

2.1 Hardware Required:

For this project, the following hardware specifications were used:

- System: Any Windows/Mac/Linux machine
- OS: Windows 11
- RAM: 8GB or more
- Processor: Intel(R) Core (TM) i5-8265U or equivalent
- Hard Disk: At least 10GB free space
- GPU: Any integrated/Nvidia/AMD GPU

2.2 Setting up Jupyter Notebook:

i. Install Python: Download and install the latest version of Python from the official website.
ii. Install Jupyter Notebook: You can install Jupyter Notebook using pip by running "pip install notebook".
iii. Install Required Libraries: Install all necessary libraries such as pandas, numpy, scikit-learn, tensorflow, and xgboost using pip. For example, run "pip install pandas numpy scikit-learn tensorflow xgboost".

3. Implementation:

3.1 Reading the Data:

```
import pandas as pd
# Load the dataset
data = pd.read_csv('online_shoppers_intention.csv')
data
```

3.2 Importing Necessary packages:

Import the necessary packages needed for the prediction, analysis, plotting, and testing:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc, mean_squared_error
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Bidirectional
import xgboost as xgb
```

3.3 Data Preparation:

Before training the models, the data needs to be cleaned and prepared:

- Handle missing values by either filling them with mean/median values or dropping the rows/columns with missing data.
- Convert categorical variables into numerical ones using encoding techniques such as one-hot encoding.

```
# Converting the categorical variables to numerical variables
label_encoder = LabelEncoder()
data['Month'] = label_encoder.fit_transform(data['Month'])
data['VisitorType'] = label_encoder.fit_transform(data['VisitorType'])
```

- Scaling the data to ensure that all features contribute equally to the model's learning process, rather than being dominated by features with larger numerical values:

```
# Scaling the data
scaler = StandardScaler()
scaled_features = scaler.fit_transform(data.drop(['Revenue'], axis=1))
```

- Split the data into training and testing sets using the train_test_split function from scikit-learn.

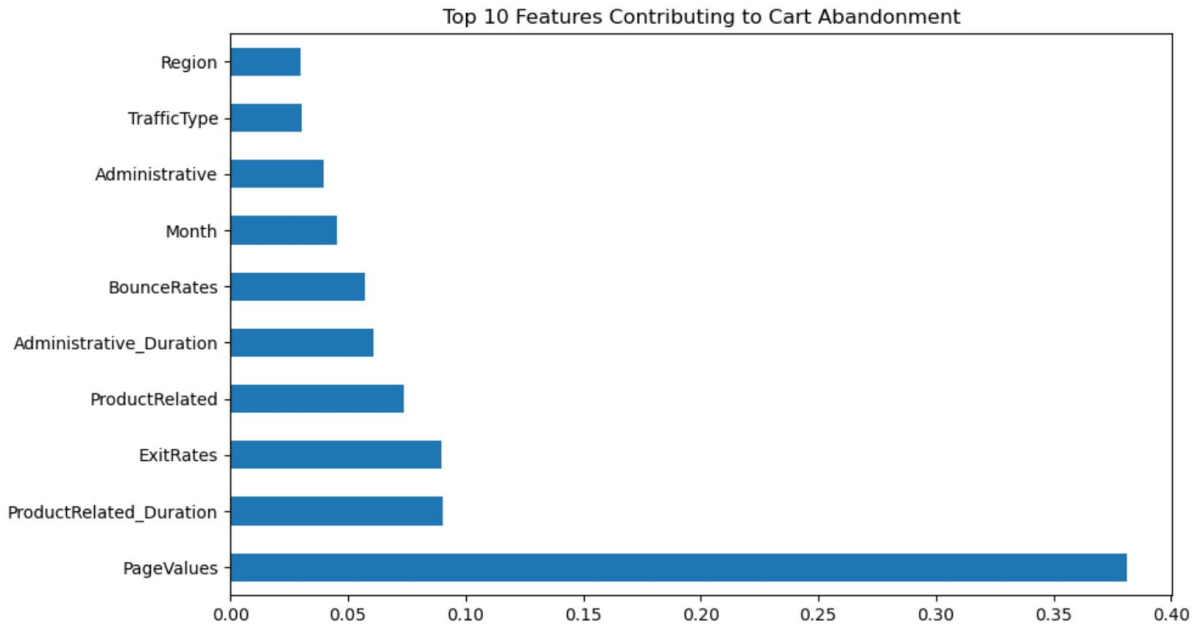
```
from sklearn.model_selection import train_test_split

X = data.drop('target_column', axis=1)
y = data['target_column']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.4 Model Training and Testing:

Different models were trained to predict cart abandonment and purchase behavior. The following are the steps for training and testing the models:

Parameter tuning:

```
# Analyze features contributing to cart abandonment
feature_importances = pd.Series(rf_model_cart.feature_importances_, index=X_cart.columns)
feature_importances.nlargest(10).plot(kind='barh', figsize=(10, 6))
plt.title('Top 10 Features Contributing to Cart Abandonment')
plt.show()
```



1. *Random Forest:*

The Random Forest model is trained by creating multiple decision trees from random subsets of the training data, then aggregating their predictions to improve accuracy and reduce overfitting.

```
# Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print('Random Forest Classifier:')
print(f'Accuracy: {accuracy_score(y_test, y_pred_rf)}')
print(classification_report(y_test, y_pred_rf))
plot_confusion_matrix(y_test, y_pred_rf, 'Random Forest')
plot_roc_curve(y_test, rf_model.predict_proba(X_test)[:, 1], 'Random Forest')
plot_feature_importance(rf_model, X_train, 'Random Forest')
```

2. *LSTM-RF:*

The LSTM-RF model combines a Long Short-Term Memory (LSTM) network to capture temporal dependencies in the data with a Random Forest classifier, which is trained on the LSTM outputs for final predictions.

```
# LSTM-RF Model (using LSTM for feature extraction)
X_train_lstm = np.expand_dims(X_train.values, axis=2)
X_test_lstm = np.expand_dims(X_test.values, axis=2)

lstm_model = Sequential()
lstm_model.add(LSTM(50, input_shape=(X_train_lstm.shape[1], X_train_lstm.shape[2])))
lstm_model.add(Dense(1, activation='sigmoid'))

lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
lstm_model.fit(X_train_lstm, y_train, epochs=10, batch_size=64, validation_split=0.2)

lstm_features_train = lstm_model.predict(X_train_lstm)
lstm_features_test = lstm_model.predict(X_test_lstm)

rf_model_lstm = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_lstm.fit(lstm_features_train, y_train)
y_pred_lstm_rf = rf_model_lstm.predict(lstm_features_test)

print('LSTM-RF Model:')
print(f'Accuracy: {accuracy_score(y_test, y_pred_lstm_rf)}')
print(classification_report(y_test, y_pred_lstm_rf))
plot_confusion_matrix(y_test, y_pred_lstm_rf, 'LSTM-RF')
plot_roc_curve(y_test, rf_model_lstm.predict_proba(lstm_features_test)[:, 1], 'LSTM-RF')
```

3. *XGBoost:*

The XGBoost model is trained using an ensemble of gradient-boosted decision trees, optimizing the model's performance by minimizing the loss function through iterative improvements.

```
# XGBoost Classifier
xgb_model = xgb.XGBClassifier(n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)

print('XGBoost Classifier:')
print(f'Accuracy: {accuracy_score(y_test, y_pred_xgb)}')
print(classification_report(y_test, y_pred_xgb))
```

4. *Bi-LSTM:*

The Bi-LSTM model is trained using a bidirectional LSTM network that processes input data in both forward and backward directions, capturing contextual information from both past and future states for better prediction accuracy.

```
# Define the Bi-LSTM model
model = Sequential()
model.add(Bidirectional(LSTM(50, return_sequences=True), input_shape=(X_train_cart_lstm.shape[1], X_train_cart_lstm.shape[2])))
model.add(Bidirectional(LSTM(50)))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train_cart_lstm, y_train_cart, epochs=10, batch_size=64, validation_split=0.2)

# Predict using the test set
y_pred_cart_lstm = (model.predict(X_test_cart_lstm) > 0.5).astype("int32")

# Evaluate the model
print('Bi-LSTM Model for Cart Abandonment:')
print(f'Accuracy: {accuracy_score(y_test_cart, y_pred_cart_lstm)}')
print(classification_report(y_test_cart, y_pred_cart_lstm))
```

3.5 Model Testing:

After training, test the models using the test data:

```
y_pred = rf_model.predict(X_test)
```

Evaluate the model's performance using metrics like accuracy, precision, recall, and F1 score.

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

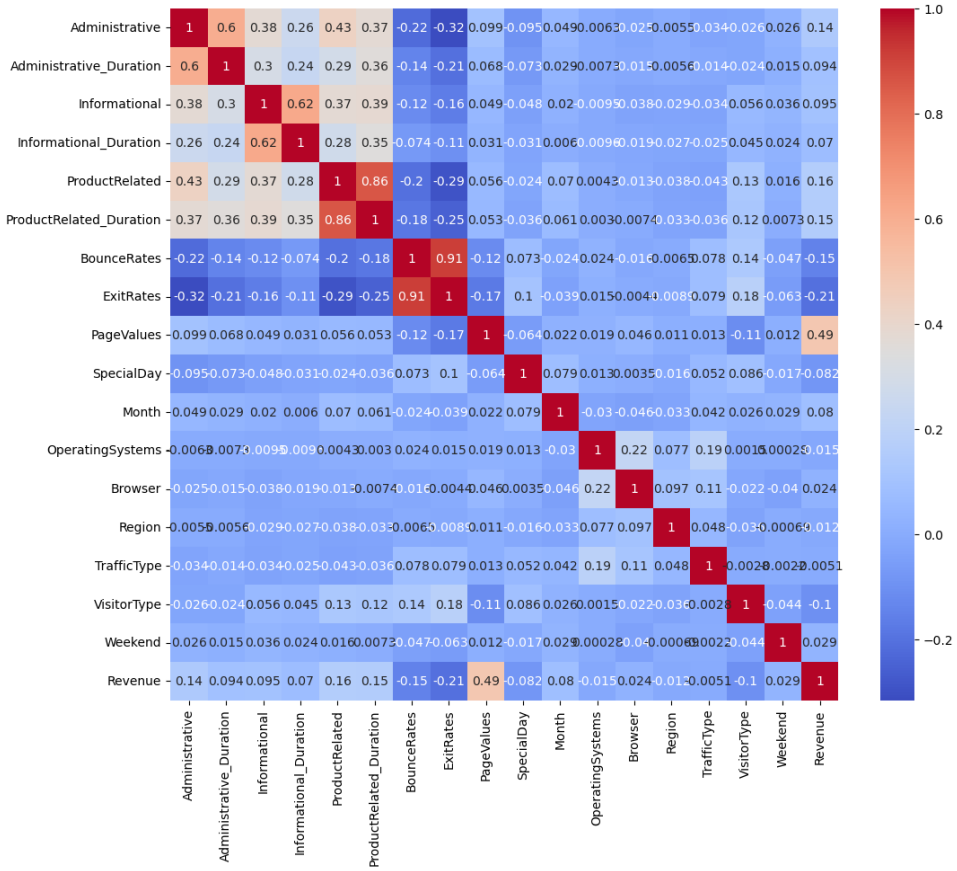
3.6 Visualisations of data:

Display of Correlation Matrix:

Input:

```
# Display the correlation matrix
corr_matrix = scaled_data.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

Output:

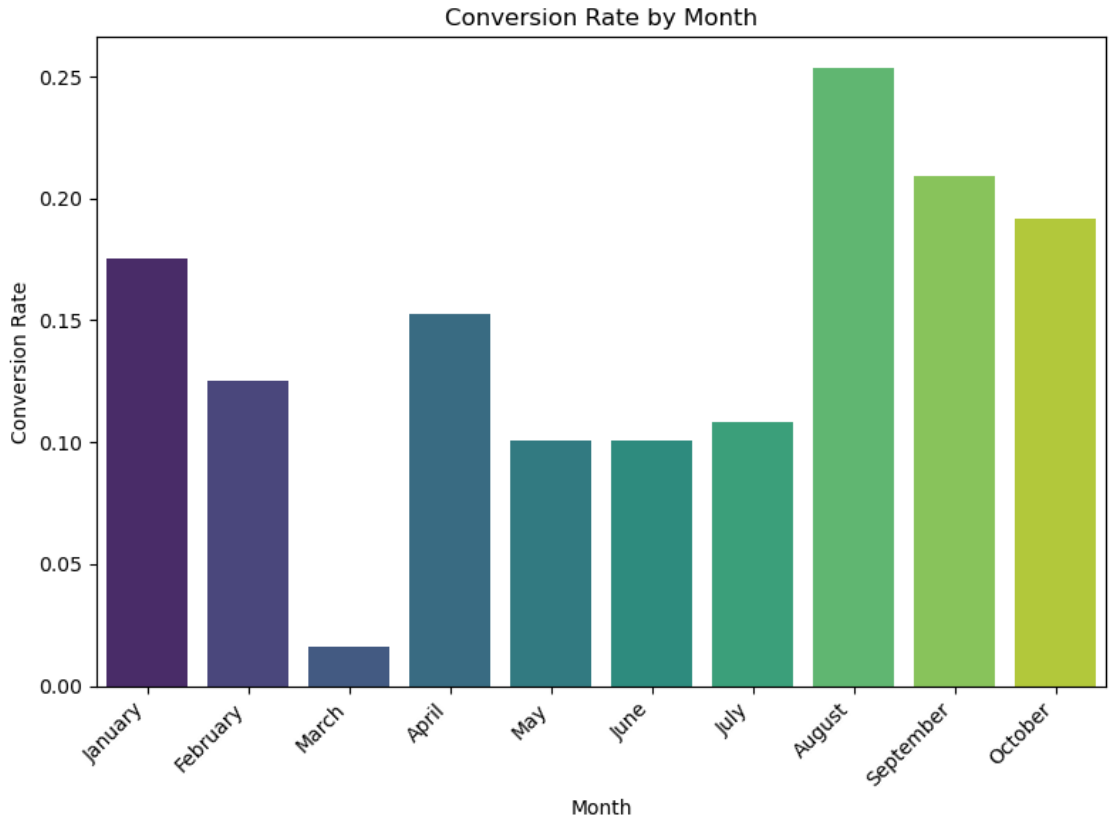


Monthly conversion rates:

Input:

```
# Map the transformed month values to month names
df['Month'] = df['Month'].map(month_names)

# Plot the conversion rates by month
plt.figure(figsize=(8, 6))
sns.barplot(x=df['Month'], y=df['Conversion_Rate'], palette='viridis')
plt.xlabel('Month')
plt.ylabel('Conversion Rate')
plt.title('Conversion Rate by Month')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Output:

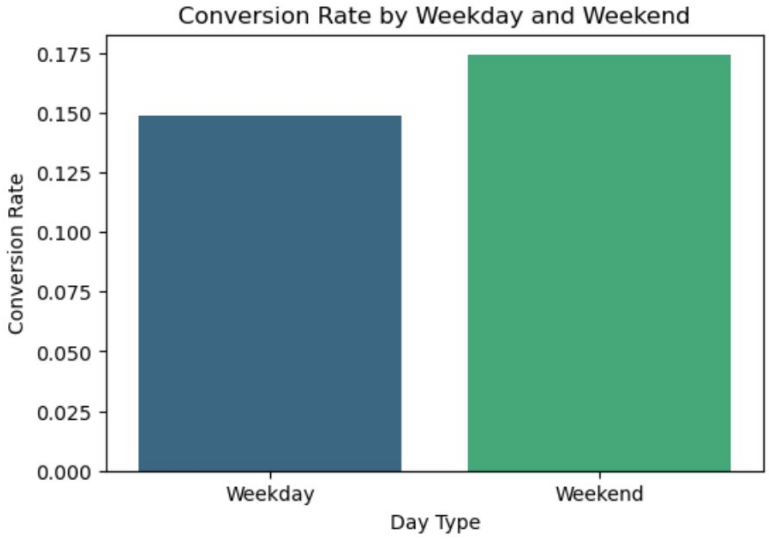


Weekend conversion rates:

Input:

```
# Map the numeric values to 'Weekday' and 'Weekend'
df_weekend['Weekend'] = df_weekend['Weekend'].map(weekend_labels)

# Plot the conversion rates by weekend/weekday
plt.figure(figsize=(6, 4))
sns.barplot(x=df_weekend['Weekend'], y=df_weekend['Conversion_Rate'], palette='viridis')
plt.xlabel('Day Type')
plt.ylabel('Conversion Rate')
plt.title('Conversion Rate by Weekday and Weekend')
plt.show()
```

Output:

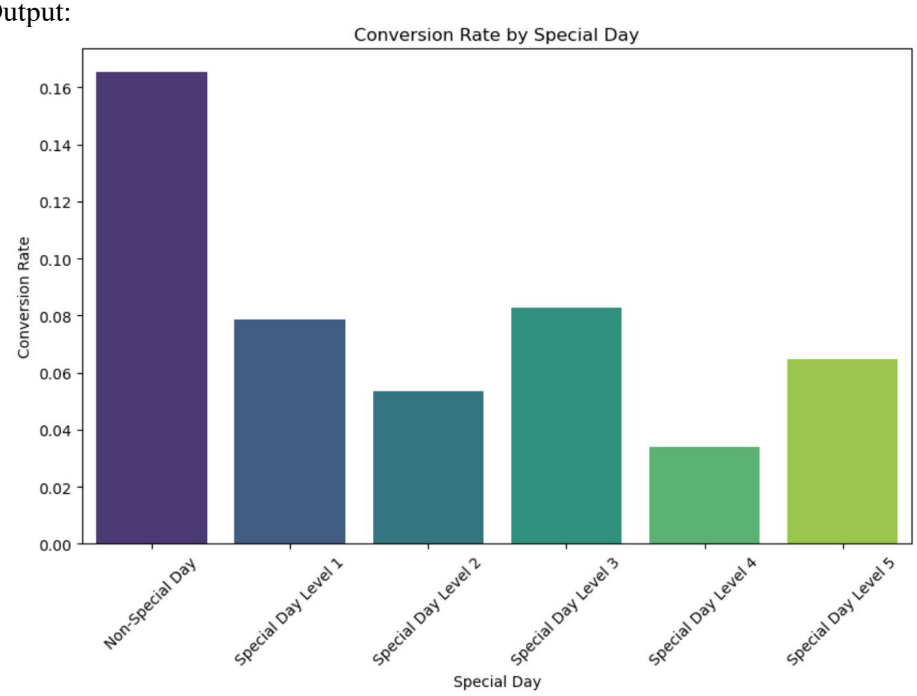


Conversion rates during special days:

Input:

```
# Map the numeric values to descriptive labels
df_special_day['SpecialDay'] = df_special_day['SpecialDay'].map(special_day_labels)

# Plot the conversion rates by special day
plt.figure(figsize=(10, 6))
sns.barplot(x=df_special_day['SpecialDay'], y=df_special_day['Conversion_Rate'], palette='viridis')
plt.xlabel('Special Day')
plt.ylabel('Conversion Rate')
plt.title('Conversion Rate by Special Day')
plt.xticks(rotation=45)
plt.show()
```

Output:



4. Conclusion

By following the instructions in this manual, users will be able to replicate the experiments, train the models on their datasets, and test them to predict cart abandonment and purchase behavior. This manual ensures that the process is straightforward and the models can be adapted or improved for future research or practical applications.

5. References

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.
2. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

3. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
4. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.
5. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).