National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Sriram Rajgopalan
Student ID: x23213876

School of Computing
National College of Ireland

Supervisor: Jaswinder Singh

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. Sriram Rajgopalan ……

Student ID: …… x23213876……..……

Programme: …… Msc in Data Analytics …… **Year:** …2024……..

Module: …… Msc Research Project ……..……

Lecturer: …… Jaswinder Singh……..……

Submission Due Date: ……12/12/2024……..……

Project Title: …Evaluating the Prevalence and Effects of Disguised Unemployment in Ireland……

Word Count: ……726…… **Page Count:** ……9……..……..……

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: ……Sriram Rajgopalan……

Date: ……12/12/2024……

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

Sriram Rajgopalan
x23213876

1 Introduction

A research project has been undertaken to predict "Prevalence of disguised Unemployment in Ireland" and this document serves as the configuration manual for the usage of various resources, steps and experiments that are discussed throughout. The research is done using a machine learning technique of "Stacking classifier" which has experimental coding, implementation and evaluation.

A complete information on the specification used to carry out the research can be found in this report. The hardware and tool specifications can be found in section 2, The data gathering and different stages of preparation and the design implementation are explained in section 3. The execution of the model is explained in section 4.

2 Hardware and Tool Specifications

• **System type:** 64-bit Windows operating system, x64-based processor

• **RAM:** 16 GB

• **Processor:** 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.70 GHz

• **Programming Language:** Python

• **Integrated Development Environment (IDE):** Jupyter Notebook

• **Python Libraries/Modules:** pandas, numpy, sklearn, matplotlib, seaborn, shap

• **Browser:** Chrome

• **Other Software:** MS Office

• **Method of execution of the code:** Code is opened in Jupyter notebook – restart kernal and run all cells would run the code.

3 Data Collection

The data used for the experiment is obtained from two different data portals ILOSTAT[1]

[1] https://ilostat.ilo.org/data/#

which are the open datasets maintained by the international labour organisation. The datasets are downloaded in the csv format and are stored locally. The datasets are renamed as "Native_Dataset_Education", "Employment_Dataset", "Education_Dataset", "Native_age_Dataset" and "Skills_Dataset". The csv is now run with python on the Jupyter notebook for further processing.

3.1 Preparation of data

Using the pandas library, the csv is imported and opened with the help of python. The structure of the data is analysed as mentioned in Figure 1 to proceed further.

```
import pandas as pd

df = pd.read_csv('Education_Dataset.csv')

print ("The rows X Columns of the dataset is:", df.shape)
print(df.head())

print(df.info())
```

```
The rows X Columns of the dataset is: (3168, 12)
  STATISTIC    Statistic Label  TLIST(Q1) Quarter C02199V02655         Sex  \
0  QLF51C01  Unemployment rate      20191  2019Q1            -  Both sexes
1  QLF51C01  Unemployment rate      20191  2019Q1            -  Both sexes
2  QLF51C01  Unemployment rate      20191  2019Q1            -  Both sexes
3  QLF51C01  Unemployment rate      20191  2019Q1            -  Both sexes
4  QLF51C01  Unemployment rate      20191  2019Q1            -  Both sexes
```

Figure 1: Data import

The empty strings are calculated on each of the column as coded in the Figure 2

```
# Count the empty strings column-wise
empty_strings_count = (ireland_df == '').sum(axis=0)

print("Count of empty strings in each column:")
print(empty_strings_count)
```

```
Count of empty strings in each column:
STATISTIC                    0
Statistic Label              0
TLIST(Q1)                    0
```

Figure 2: Check for empty strings

The unimportant columns which are column descriptors/ repetetive labels are removed as per the Figure 3.

```
columns_to_remove = ['STATISTIC',   'Statistic Label',  'TLIST(Q1)', 'C02199V02655', 'C02076V03371', 'C04283V05060','UNIT'
]

# Remove the columns from the dataframe
ireland_df = ireland_df.drop(columns=columns_to_remove)

# Confirm the changes
print(ireland_df.head())
```

```
  Quarter         Sex Age Group  \
```

Figure 3: Removal of descriptor columns

The Labels are classified as per ILOSTAT standards for better understanding as mentioned in the Figure. Refer Figure 1 for the classification matrix of all the datasets.

Education Level Classifiers

S.No.	Label	Classified Keyword
1	Levels 1-2	Primary and lower secondary education
2	Levels 3 and 4	Upper secondary and post-secondary non-tertiary education
3	Levels 5-8 (Levels 0-8)	Tertiary education

Employment Skill Classifiers

S.No.	Label	Classified keyword
1	Managers	Skill levels 3 and 4 ~ high
2	Technicians and associate professionals	Skill levels 3 and 4 ~ high
3	Professionals	Skill levels 3 and 4 ~ high
4	Clerical support workers	Skill level 2 ~ medium
5	Service and sales workers	Skill level 2 ~ medium
6	Craft and related trades workers	Skill level 2 ~ medium
7	Plant and machine operators, and assemblers	Skill level 2 ~ medium
8	Elementary occupations	Skill level 1 ~ low
9	Armed Forces Occupations	Skill level 1 ~ low

Employment Type Classifiers

S.No.	Label	Classified Keyword
1	In employment part-time - underemployed	Disguised Unemployment
2	In employment part-time - not underemployed	Disguised Unemployment
3	Unemployed seeking full-time work/future job-starter	Disguised Unemployment
4	Unemployed seeking part-time work	Disguised Unemployment
5	Potential additional labour force	Disguised Unemployment
6	In labour force	Relatively Employed
7	All ILO economic status	Relatively Employed
8	In employment	Relatively Employed
9	In employment full-time	Relatively Employed
10	In employment part-time	Relatively Employed
11	Unemployed	Unemployed
12	ILO Economic Status	Unemployed
13	Not in labour force	Unemployed
14	Others not in labour force	Unemployed
15	Unemployed seeking work as self-employed	Unemployed

Classify Labels as per ILOSTAT

[6]:
```
def replace_with_skill_level(label):
    label = label.strip()
    if any(keyword in label for keyword in ["Levels  0-8", "Levels 5-8"]):
        return "Tertiary education"
    elif any(keyword in label for keyword in ["Levels 1-2"]):
        return "Primary and lower secondary education"
    else:
        return "Upper secondary and post-secondary education"

ireland_df['Education Attainment Level'] = ireland_df['Education Attainment Level'].apply(replace_with_skill_level)


unique_values = ireland_df['Education Attainment Level'].unique()

print(ireland_df)
print("Unique Values:", unique_values)
```

```
     Quarter         Sex      Age Group  \
```

Figure 3: Labeling as per ILOSTAT

Post classification the datatypes are checked for strings and are changed into numerical as mentioned in the Figure 4

```
[8]: print(ireland_df.dtypes)

     # Identify all categorical (non-numeric) columns
     categorical_columns = ireland_df.select_dtypes(include=['object']).columns

     # Exclude 'classif2.label' from the list of categorical columns
     categorical_columns = categorical_columns.drop('Education Attainment Level')

     # Display the final list of categorical columns
     print("Categorical Columns (excluding 'Education Attainment Level'):", categorical_columns)
```

```
Quarter                      object
Sex                          object
Age Group                    object
Education Attainment Level   object
VALUE                       float64
dtype: object
Categorical Columns (excluding 'Education Attainment Level'): Index(['Quarter', 'Sex', 'Age Group'], dtype='object')
```

Figure 4: Check for Data types

The categorical features are encoded before runing the model using label encoder as mentioned as Figure 5.

```
from sklearn.preprocessing import LabelEncoder

# Apply LabelEncoder to each categorical column
for col in categorical_columns:
    le = LabelEncoder()
    ireland_df[col] = le.fit_transform(ireland_df[col])

print("Encoded Data:")
print(ireland_df)
```

```
Encoded Data:
      Quarter  Sex  Age Group                    Education Attainment Leve
0           0    0          3                             Tertiary educati
```

Figure 5: Label Encoder

3.2 Design Implementation of data

The model consists of 4 sections, Data collection, Feature engieering , Training of the model and results and interpretation. The model architecture is depicted in the Figure 6.

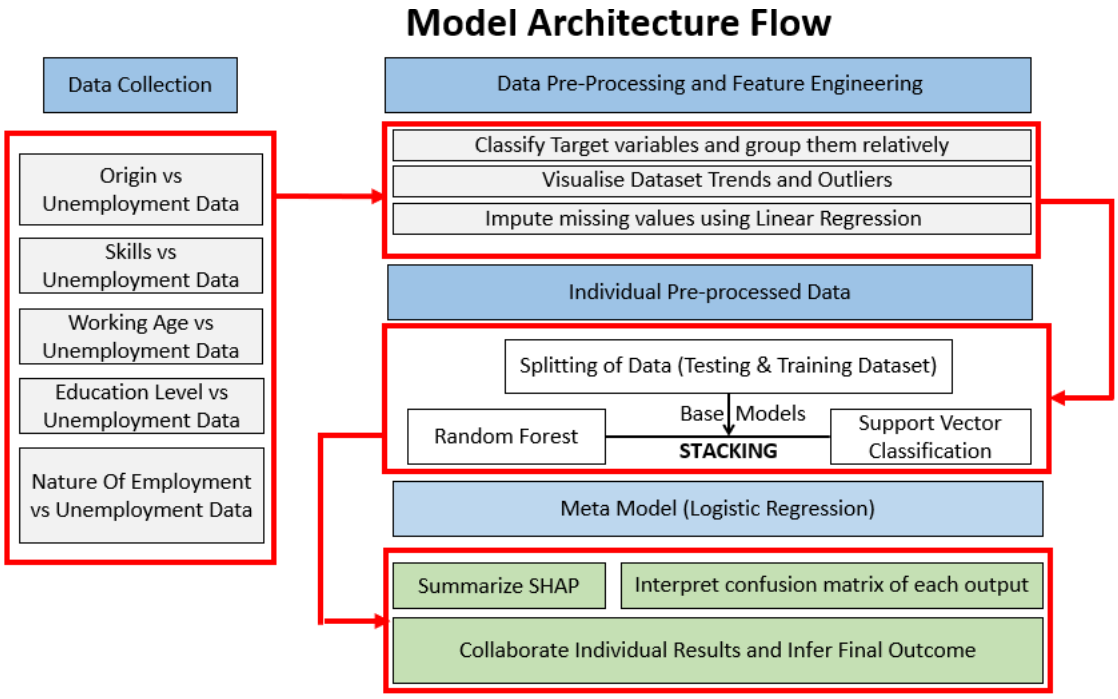

Figure 6: Model Architecture

The individual pre processed data is led into the training of the base models of the "Stacking Classifier" that consists of Random Forest and Support Vector Classification. A sample pipeline is mentioned in the Figure 7.

Once the model enters the pipeline, the output of base classifier is set into the Meta model and final prediction is done. This process is repeated for all 5 datasets as seen on Figure 8

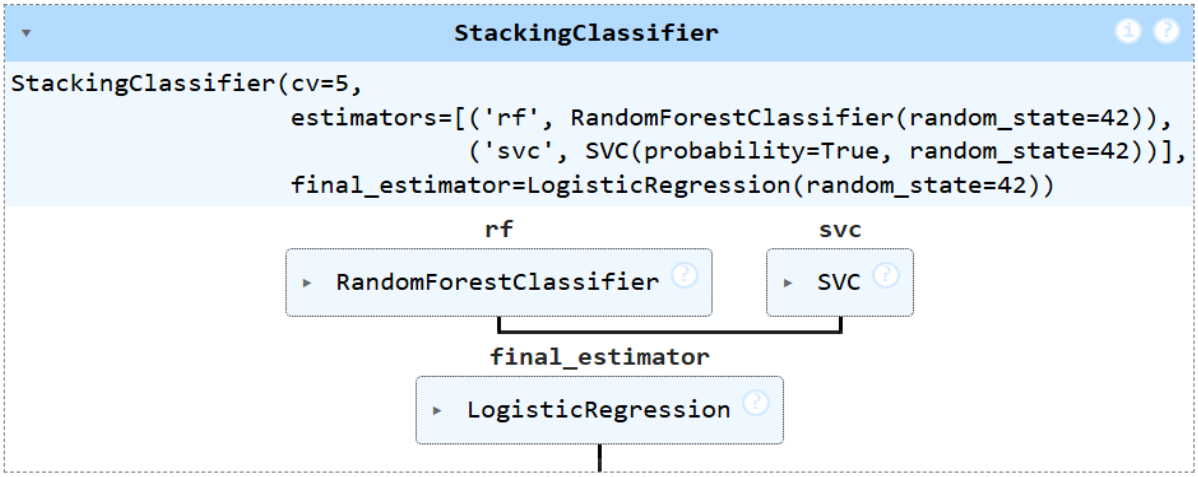

Figure 7: Stacking Classifier Pipeline

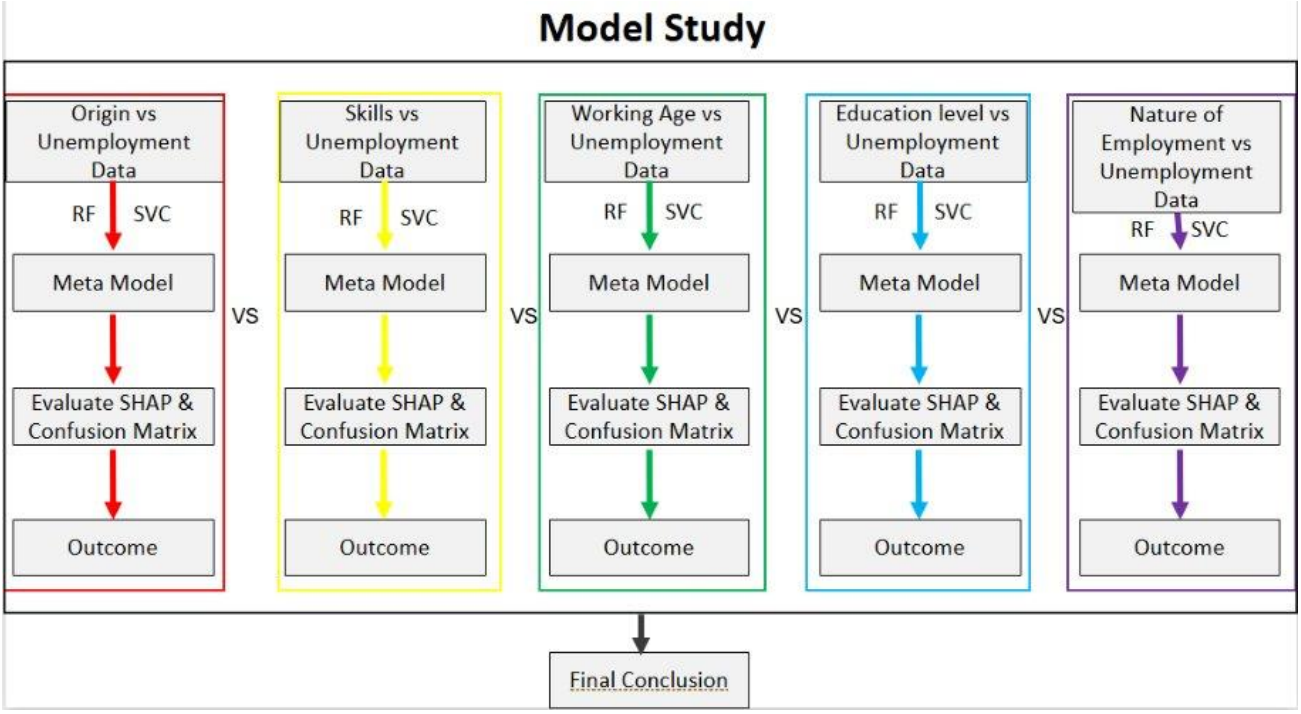

Figure 8: Model Study

4 Model Execution

The pre-processed data is split into 80:20 and the same is modelled into the base learners using RF and SVC and then the meta model Logistic Regression is learning from the output of the base learners as explained in the Figure 9.

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

# Initialize base models
estimators = [
    ('lr', LogisticRegression()),
    ('rf', RandomForestClassifier()),
    ('svc', SVC(probability=True))
]

# Meta-model
meta_model = LogisticRegression()
```

Figure 9: Testing and Training Model

The stacking classifier is run as shown in the Figure 10 on the meta model and classification report is taken as the first part of the metric.

```
# Create Stacking Classifier
stacking_clf = StackingClassifier(estimators=estimators, final_estimator=meta_model)


# Train and evaluate
stacking_clf.fit(X_train, y_train)
y_pred = stacking_clf.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)

print(f"Stacking Classifier Accuracy: {accuracy}")
print(classification_report(y_test, y_pred))
```

Figure 10: Initiation of Stacking classifier

Since this a multiclass classification, the test labels are binarized and a ROC-AUC curve is plotted as mentioned in the Figure 11. This is the second metric to test.

```
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Assume class_names is a list of integer-encoded classes, e.g., [0, 1, 2]
# If the class labels are encoded as integers, we ensure that label_binarize knows all class labels

# Binarize the true labels and predicted labels
y_test_bin = label_binarize(y_test, classes=range(len(class_names)))
y_pred_bin = label_binarize(y_pred, classes=range(len(class_names)))

# Initialize dictionaries to store the FPR, TPR, and AUC for each class
fpr = {}
tpr = {}
roc_auc = {}
legend_labels = []
```

Figure 21: Binarization for roc-auc curve

The encoded Labels are decoded, and a confusion matrix is created in order to analyse the classification of classes as in Figure 12

```
# Decode the labels back to the original values
y_test_decoded = label_encoder.inverse_transform(y_test)
y_pred_decoded = label_encoder.inverse_transform(y_pred)

# True labels (actual class names) and predicted labels
class_names = label_encoder.classes_  # Get the class names from the encoder

# Create confusion matrix
cm = confusion_matrix(y_test_decoded, y_pred_decoded)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Figure 12: Label Decoder and confusion matrix

Finally a feature importance score is calculated and SHAP values are plotted as in Figure 13 in order to conclude the objective of the project with the feature interaction.

```
from sklearn.inspection import permutation_importance

# Train any model (e.g., Stacking Classifier)
stacking_clf.fit(X_train, y_train)

# Calculate Permutation Importance
perm_importance = permutation_importance(stacking_clf, X_test, y_test, n_repeats=10, random_state=42)
importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': perm_importance.importances_mean
}).sort_values(by='Importance', ascending=False)

print(importance_df)
```

```
# Compute SHAP values for test data
shap_values = explainer.shap_values(X_test)

# --- SHAP Plots ---

# Summary Plot
shap.summary_plot(shap_values, X_test)
```

Figure 33: Feature Importance and SHAP