Configuration Manual

MSc Research Project
Data Analytics

Vinay Babu Gundu
Student ID: x23228491

School of Computing
National College of Ireland

Supervisor: Prof. Anu sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vinay babu gundu
Student ID:	X23228491
Programme:	MSc in Data analytics **Year:** 2024 - 25
Module:	MSc Research Project
Lecturer:	Prof.Anu sahni
Submission Due Date:	29-1-2025
Project Title:	House price prediction in Beijing

Word Count:854 **Page Count:** 8

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: Vinay babu gundu

Date: 29-1-2025

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

Vinay Babu Gundu

Student ID: x23228491

1. Introduction

This configuration guide details the reproduction of the experiment setup and results of this study on fake news identification by a dual Optical Character Recognition (OCR) and multimodal deep learning framework. The system incorporates text and image analysis in the detection of fake news with high precision through state-of-the art OCR, BERT based transformer and CNN-based ResNet models. This document has all details about the packages and software used, coupled with all configurations required that ensures this system provides experimental environment thereby similar results.

2. Deployment Environment

2.1 Hardware Specification

- **Processor:** Intel Core i7 or equivalent
- **RAM:** 16 GB or higher
- **GPU:** NVIDIA RTX 2060 or higher (recommended for training ANN).

2.2 Software Specification

- **Operating System:** Windows 10/11, macOS, or Linux-based OS
- **Programming Language:** Python 3.11
- **IDE:** Jupyter Notebook or VS Code (with Python extension)

2.3 Python Libraries Required

Figure 2 shows the list of the necessary Python Libraries required for the execution of the code. This mentioned python libraries can be installed using the pip command.

- **pandas:** For data manipulation and analysis.
- **NumPy:** For numerical operations, especially array operations.

Data Visualization Libraries:

- **Seaborn:** For statistical data visualization.

- **Matplotlib:** For creating static, animated, and interactive visualizations.

Machine Learning Libraries:

- **scikit-learn:** For various machine learning algorithms, including:
  - Linear Regression ○ Random Forest Regression ○ Support Vector Regression (SVR) ○
  - Decision Tree Regression ○
  - XGBoost Regression ○
  - Gradient Boosting Regression
- **XGBoost:** For gradient boosting algorithms. **Other Libraries:**
- **warnings:** For filtering out warning messages.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
import warnings
warnings.filterwarnings("ignore")
```

Figure 1: Libraries Imported

3. Data Source

The dataset for this project contains historical house pricing data, including features like location, size, number of rooms, and other property-related characteristics.

Link: **https://github.com/teja0508/House-Price-Prediction-in-Beijing**

Steps to Prepare the Dataset: 1.

Data Source:

- ○ The dataset should include both numerical and categorical features relevant to house price prediction. ○ If using a public dataset, ensure it has been cleaned and formatted appropriately.

2. **Data Pre-processing Techniques:**

- ○ Handle missing values using imputation. ○ Normalize or standardize numerical features to improve model training stability. ○ Encode categorical variables using one-hot or label encoding.

```
# Check for missing values in each column to understand data completeness
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0].sort_values(ascending=False)

# Display columns with missing values
missing_values
```

```
DOM                    157977
buildingType             2021
communityAverage          463
elevator                   32
fiveYearsProperty          32
subway                     32
dtype: int64
```

The dataset contains missing values in the following columns:

DOM: 157,977 missing values buildingType: 2,021 missing values communityAverage: 463 missing values elevator, fiveYearsProperty, and subway: each with 32 missing values

```
# Check data types to understand columns that might need conversion or cleaning
data_types = df.dtypes
data_types
```

Figure 2: Handling missing values and cleaning

4. Project Code Files

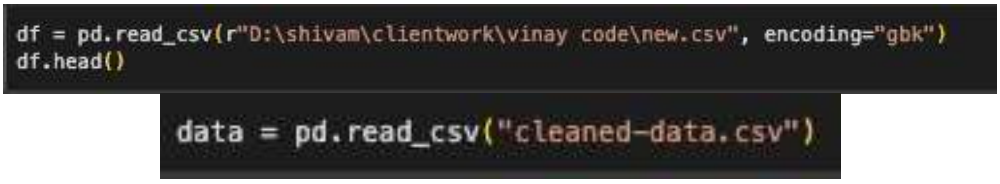
Main Colab Notebooks:

1. **Data Preprocessing:** Handles data cleaning, encoding, and feature scaling.
2. **Model Training:** Includes the implementation of Linear Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost, and ANN.
3. **Performance Evaluation:** Calculates evaluation metrics for each model.
4. **Results Visualization:** Compares model performance using graphs and charts.

5. Data Preparation

5.1 Extracting Data

Loading the datasets from CSV file uploaded:



```
df = pd.read_csv(r"D:\shivam\clientwork\vinay code\new.csv", encoding="gbk")
df.head()
```

```
data = pd.read_csv("cleaned-data.csv")
```

Figure 3: Loading the datasets

5.2 Data Pre-processing

- Handling Missing Values: Impute missing data using mean/median or interpolate.
- Data Separation: separating the data variables**.**
- This format is repeated for every model building code as well as EDA.

```
# Check for missing values in each column to understand data completeness
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0].sort_values(ascending=False)

# Display columns with missing values
missing_values
```

```
DOM                  157977
buildingType           2021
communityAverage        463
elevator                 32
fiveYearsProperty        32
subway                   32
dtype: int64
```

The dataset contains missing values in the following columns:

DOM: 157,977 missing values buildingType: 2,021 missing values communityAverage: 463 missing values elevator, fiveYearsProperty, and subway: each with 32 missing values

```
# Check data types to understand columns that might need conversion or cleaning
data_types = df.dtypes
data_types
```

Figure 4: To handle display missing values

6. Model Building

Models used:

- Linear Regression
- Decision Tree Regressor
- Random Regressor
- Gradient Boosting
- XGBoost Regressor
- ANN

```
# Prepare features and target
X = data.drop(columns=['totalPrice'])
y = data['totalPrice']

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

# Initialize models with GPU support where applicable
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(random_state=42),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'XGBoost Regressor': XGBRegressor(tree_method='gpu_hist', random_state=42),  # Enable GPU
    'Gradient Boosting': GradientBoostingRegressor(random_state=42)
}

# Train and evaluate each model
results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results[model_name] = {'Mean Squared Error': mse, 'R^2 Score': r2}

# Display results
results_df = pd.DataFrame(results).T
results_df
```

Figure 5: Model training code snippet for models

```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define the ANN model with additional layers and dropout for regularization
# Define the model
model = Sequential()

# Input layer and first hidden layer
model.add(Dense(units=64, activation='relu', input_shape=(X_train.shape[1],)))

# Additional hidden layers
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=16, activation='relu'))

# Output layer
model.add(Dense(units=1))  # Single output for regression (no activation function)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error', metrics=['mae'])

# Train the model with early stopping
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32)

# Predict on test data
y_pred = model.predict(X_test)

# Evaluate the model on the test set
test_loss, test_mae = model.evaluate(X_test, y_test)
print(f"Mean Absolute Error on test set: {test_mae}")

# Calculate R2, RMSE, and MAE
# Calculate R-squared
r2 = r2_score(y_test, y_pred)
print(f"R-squared: {r2}")

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"RMSE: {rmse}")

# Plot training & validation loss
import matplotlib.pyplot as plt

# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.show()
```

Figure 6: ANN model code snippet

7.3 Evaluation

Metrics Calculated:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- R^2 Score

	Mean Squared Error	R^2 Score
Linear Regression	12641.984726	0.766750
Random Forest	3292.259661	0.939256
Decision Tree	5498.519372	0.898550
XGBoost Regressor	2873.271525	0.946987
Gradient Boosting	4687.166409	0.913520

```
Mean Absolute Error on test set: 33.525447845458984
R-squared: 0.9344106591269139
RMSE: 59.622885225720196
```

Figure 7: Results for Models

7. Results and Visualizations

- **XGBoost Regressor:**
- Achieved MSE of 2,873.27 and R^2 of 0.947, indicating its ability to capture complex patterns.
- **Artificial Neural Network (ANN):**
- Demonstrated flexibility with MAE = 33.53 and R^2 = 0.9344.
- **Other Models:**
- Linear Regression and Decision Tree struggled with nonlinear interactions, while Random Forest and Gradient Boosting provided moderate performance.

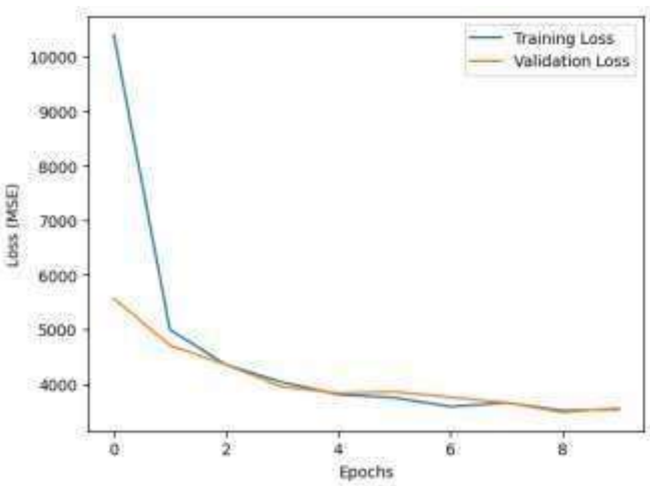


Figure 8: Training validation loss with epochs running comparison

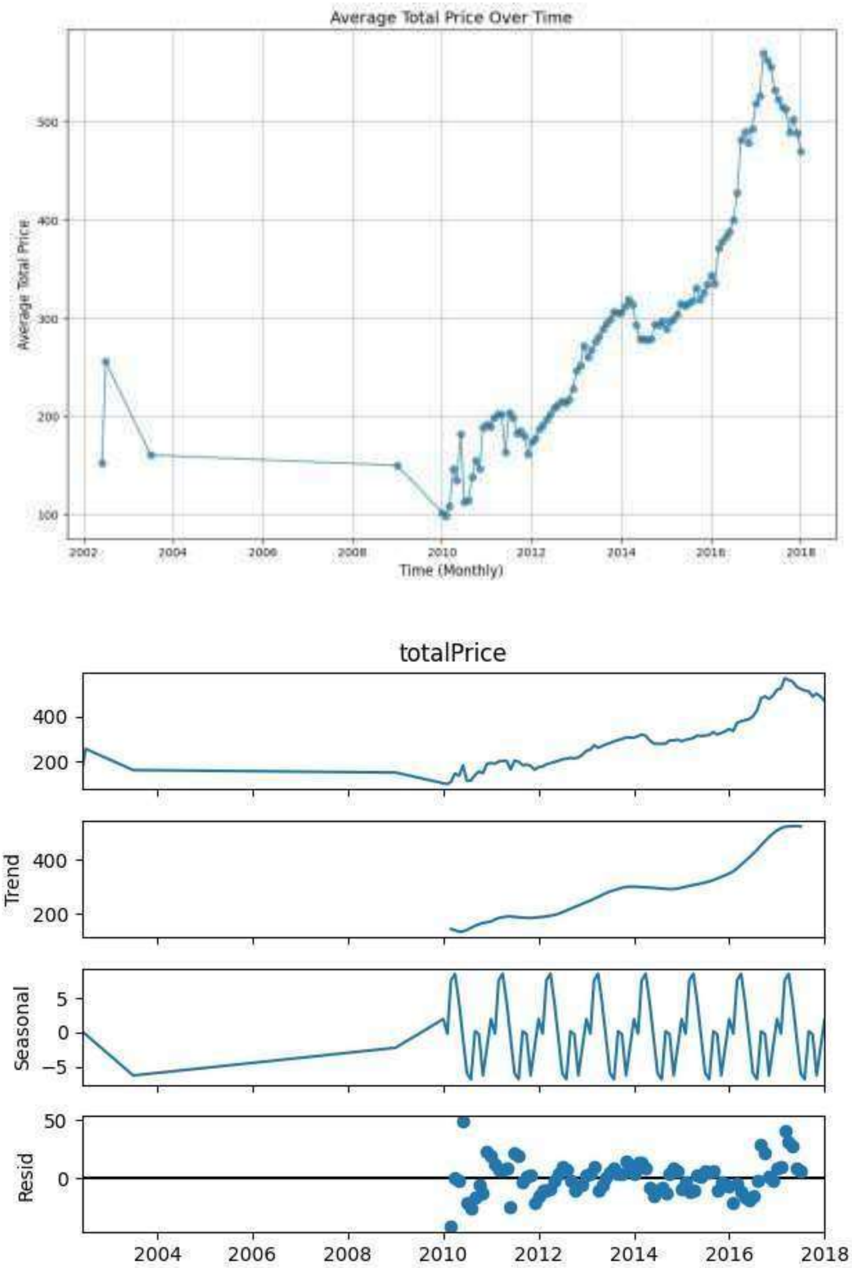


Figure 9: Resulting visualization for Time Series Analysis