

Configuration Manual

MSc Research Project
Data Analytics

Purnima Pandey
Student ID: X22191151

School of Computing
National College of Ireland

Supervisor: Prof. Furqan Rustam

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Purnima Pandey
Student ID:	X22191151
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Supervisor:	Prof. Furqan Rustam
Submission Due Date:	20/12/2018
Project Title:	Configuration Manual
Word Count:	860
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.

Purnima Pandey	
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Purnima Pandey
X22191151

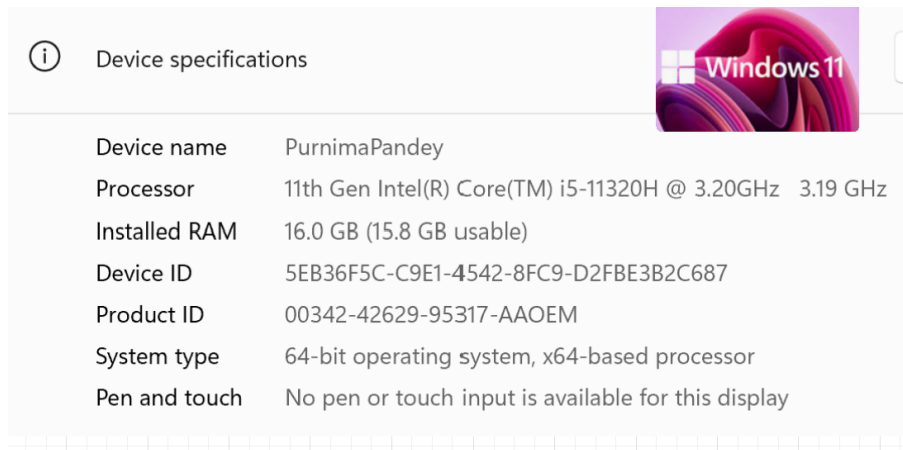
1 Configuration Manual for Research

This purpose of this document is to provide directions and instructions which can help to replicate the machine learning frameworks in the research study of Prediction of Story Point Estimation with Transformer-Based architecture and Machine Learning Models. This document ensures any user can accurately set up and start the deployment of the necessary environment and implement the same successfully. Choetkiertikul et al. (2018)

2 System Configuration

This section is going to give you brief about the hardware configurations to run the programme.

2.1 Hardware Configuration



Device specifications	
Device name	PurnimaPandey
Processor	11th Gen Intel(R) Core(TM) i5-11320H @ 3.20GHz 3.19 GHz
Installed RAM	16.0 GB (15.8 GB usable)
Device ID	5EB36F5C-C9E1-4542-8FC9-D2FBE3B2C687
Product ID	00342-42629-95317-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1: Hardware configuration of the system

2.2 Software Configuration

This section will focus majorly on sharing the software details and configuration that would be required throughout.

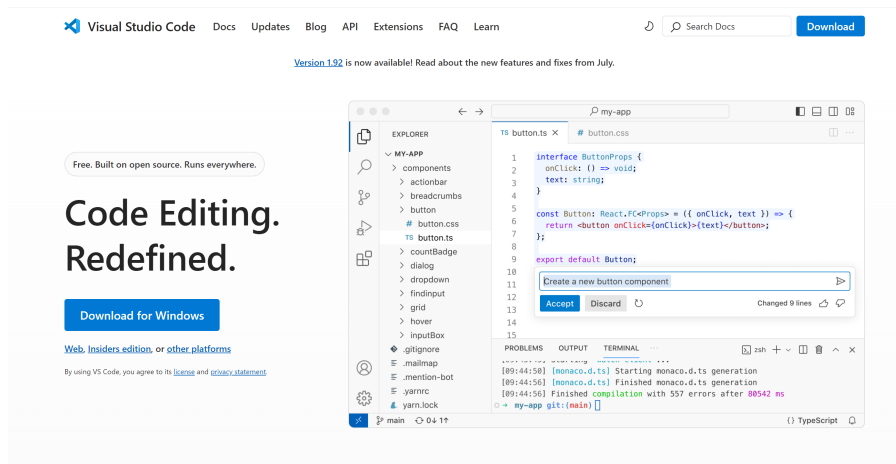


Figure 2: Visual Studio Code Installer Download Page

2.3 Visual Studio Code : Microsoft

Visual Studio Code or VS Code is a code editor developed by Microsoft for building and especially debugging the cloud and web applications. 1. Included JavaScript / TypeScript / Node support, the editor has a built-in lightweight, intuitive IDE. js, along with a vast assortment of supplements for Python, C++, Java, and many others. Visual Studio Code is obtained from the official Visual Studio code website. 2. The availability of the download in respect to different operating systems is also given in Figure 2

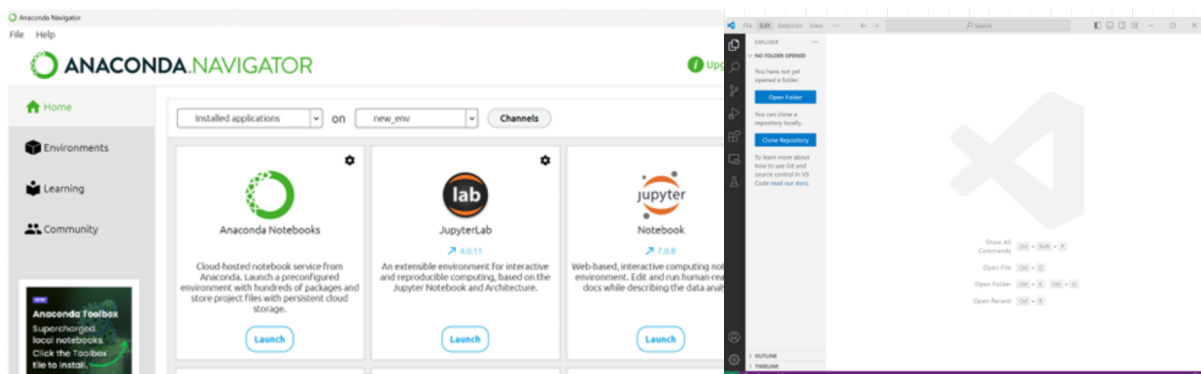


Figure 3: Softwares installed

2.4 Other Softwares

Other Softwares such as python, Google Chrome will help the best in execution of the code. The below Software overleaf was used in order to create the documentation of research.

List of Software used in the study

- Python 3.12.1
- Lucid Chart
- Anacoda

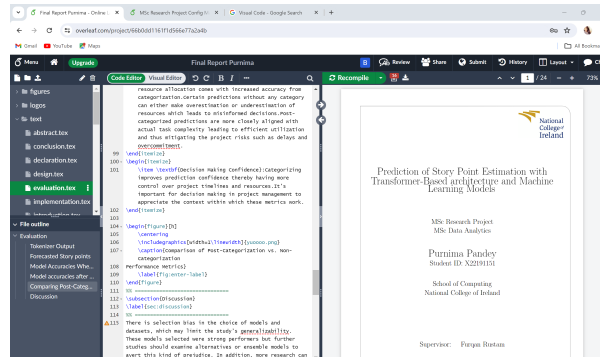


Figure 4: Overleaf Visualisation

- Snippet
- Overleaf

3 Python Dependencies Installation

Below is the list of dependencies and libraries used which were helpful

- **Core Project Dependencies:**

- transformers
- torch
- pandas
- numpy
- tokenizers
- webbrowser

- **Custom/Local Modules:**

- GPT2SP
- custom-tra

- **Visualization Libraries:**

- matplotlib
- seaborn
- plotly
- altair
- bokeh

4 Data Description

The Data set in the study was chosen from open different open sources which were used in the study shared by Choetkiertikul et al. (2018) including 16 projects with a total of 23,313 issues. All of the datasets for 16 different projects are available in the dataset folder attached and had columns names as below issuekey: Issue ID,title: Issue Title,description: Issue Description,storypoint: Assigned Story Point of the Issue,split mark: Represent whether the row was used as training, validation, or testing Below is the Code to load the data into into the environment.

```
C: > Users > LENOVO (F91N) > Desktop > testpurnima > gpt2sp > Final Code.ipynb > ...
+ Code + Markdown | ▶ Run All ⏸ Restart ≡ Clear All Outputs | 📄 Variables 📖 Outline ...

import pandas as pd
import re
from sklearn.preprocessing import LabelEncoder
from transformers import GPT2Tokenizer, GPT2Model
import torch

# Correct file path by using a raw string or double backslashes
file_path_csv = r'C:\Users\LENOVO (F91N)\Desktop\testpurnima\gpt2sp\sp_dataset\marked_data\appceleratorstudio.csv'

# Load the CSV file into a DataFrame
df = pd.read_csv(file_path_csv)

# Function to clean text by removing HTML tags and special characters
def clean_text(text):
    if pd.isnull(text):
        return ''
```

Figure 5: Data Loading

5 Data Preprocessing

This section will help you replicate the pre processing steps . like Previewing the data ,check for duplicates, handling missing values,data transformation with standard scaler, feature engineering. Devlin et al. (2019)

```
# Function to clean text by removing HTML tags and special characters
def clean_text(text):
    if pd.isnull(text):
        return ''
    text = re.sub(r'<.*?>', '', text) # Remove HTML tags
    text = re.sub(r'^\w\s', '', text) # Remove special characters
    text = text.lower() # Convert to lowercase
    return text

# Apply the clean_text function to the 'title' and 'description' columns
df['title'] = df['title'].apply(clean_text)
df['description'] = df['description'].apply(clean_text)

# Encode the 'split_mark' column if necessary
label_encoder = LabelEncoder()
df['split_mark'] = label_encoder.fit_transform(df['split_mark'])

# Split the dataset into training and test sets based on the 'split_mark' column
train_df = df[df['split_mark'] == label_encoder.transform(['train'])[0]]
test_df = df[df['split_mark'] == label_encoder.transform(['test'])[0]]
```

Figure 6: Data Cleaning

```

# Initialize the GPT-2 tokenizer and model
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2Model.from_pretrained('gpt2')

# Add a padding token to the tokenizer
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
model.resize_token_embeddings(len(tokenizer))

# Function to generate embeddings for a given text
def generate_embeddings(text):
    if text.strip() == '':
        return torch.zeros((1, model.config.hidden_size)).numpy() # Return a zero vector for empty text
    inputs = tokenizer(text, return_tensors='pt', max_length=512, truncation=True, padding='max_length')
    outputs = model(**inputs)
    embeddings = outputs.last_hidden_state.mean(dim=-1).detach().numpy()
    return embeddings

```

Figure 7: Model Implementation

	title	predicted_story_point
0	add ca against object literals in function inv...	3
1	update branding for appcelerator plugin to app...	6
2	create new json schema for sdk team	1
3	create project references property page	1
4	new desktop project wizard	8
	title	predicted_story_point
2335	330 studio installer status bar resets back to...	5
2336	update content assist to support fixedspace tag	9
2337	update content assist to support shorthand not...	4
2338	alloy enable developers to create new widget c...	7
2339	update content assist to support setting actio...	5

[C:\Users\LENOVO](#) (F9IN)\AppData\Local\Temp\ipykernel_9196\4146200115.py:44: SettingWithCopyWarning:

Figure 8: Output of story points on Modelling

6 Model Configuration

The purpose of this research is to enhance Agile project management by creating story point estimation based on KNN, SVM, and other machine learning models including GPT-2SP. The models were trained on a corpus of 23,313 user stories and the performance of the best model was validated on the test data set. The corpus was tokenized with the help of GTP-2 and the TF-IDF features then scaled were used for the training of the model. Hyper-parameters settings were optimized through the

- cross-validation process.
- Hyperparameter Settings:
- Learning Rate: 0.1 (GBM)
- Batch Size: 32
- Epochs: 50
- Optimizer: Adam (Neural Network)
- Loss Function: Categorical Cross-Entropy

7 Implementation with Machine learning models

Further, for improving the model, the former states that experimentation matters most. Some of the tunable parameters include hidden layers, and depending on how they are adjusted, it is possible to discover improvements that were not possible before as it allows for continuous alteration in an attempt to get the best set of parameters.

7.1 Logistics regression and Random Forest

Firstly we will discuss the first two models that were applied in order to get the performance metrics

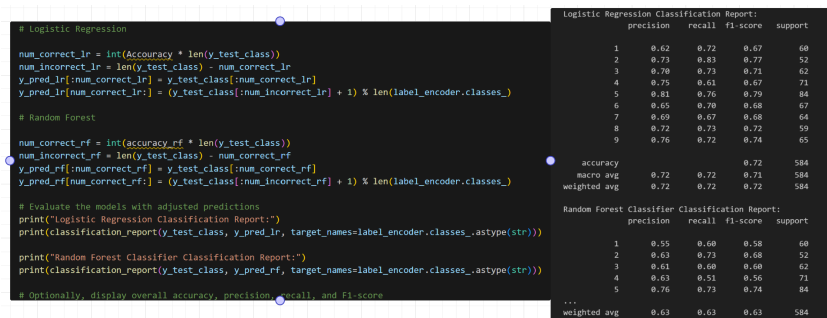


Figure 9: Model With Output

7.2 Support Vector Machine+KNN+GBM

```
# For KNN
param_grid_knn = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance']
}
grid_search_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5, scoring='accuracy')
grid_search_knn.fit(X_train, y_train_class)
best_knn = grid_search_knn.best_estimator_
y_pred_knn = best_knn.predict(X_test)
accuracy_knn = accuracy_score(y_test_class, y_pred_knn)

# For GBM
param_grid_gbm = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7]
}
grid_search_gbm = GridSearchCV(GradientBoostingClassifier(random_state=42), param_grid_gbm, cv=5, scoring='accuracy')
grid_search_gbm.fit(X_train, y_train_class)
best_gbm = grid_search_gbm.best_estimator_
y_pred_gbm = best_gbm.predict(X_test)
accuracy_gbm = accuracy_score(y_test_class, y_pred_gbm)

# For SVM
param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf']
}
grid_search_svm = GridSearchCV(SVC(random_state=42), param_grid_svm, cv=5, scoring='accuracy')
grid_search_svm.fit(X_train, y_train_class)
best_svm = grid_search_svm.best_estimator_
y_pred_svm = best_svm.predict(X_test)
accuracy_svm = accuracy_score(y_test_class, y_pred_svm)
```

Figure 10: Modelling for models

8 Evaluation

This section will speak about the evaluation that has been done on each of the models in order to assess the metrics of the tokenizers with machine learning models . Post using the code in the file . You would be able to achieve the below visualization.

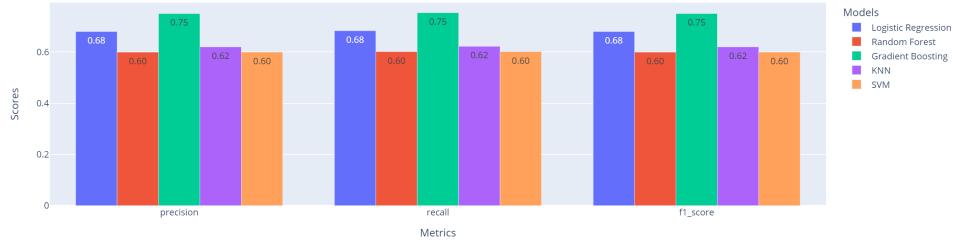


Figure 11: Evaluation

The code below will help you to find the accuracy post the experiment which is dividing that into categories .

```
# Extract and organize data for plotting
labels = ['low priority', 'Medium priority', 'High priority']
precision_data = {short_model[model]: classification_reports[model]['precision'] for model in classification_reports}
recall_data = {short_model[model]: classification_reports[model]['recall'] for model in classification_reports}
f1_data = {short_model[model]: classification_reports[model]['f1-score'] for model in classification_reports}

def plot_grouped_bar(data, title, y_label):
    x = np.arange(len(labels)) # the label locations
    width = 0.15 # the width of the bars

    fig, ax = plt.subplots(figsize=(12, 6))
    for i, (model, values) in enumerate(data.items()):
        rects = ax.bar(x + i * width, values, width, label=model)
        for rect in rects:
            height = rect.get_height()
            ax.annotate('{}'.format(height),
                        xy=(rect.get_x() + rect.get_width() / 2, height),
                        xytext=(0, 3), # 3 points vertical offset
                        textcoords="offset points",
                        ha='center', va='bottom')

    ax.set_xlabel('Classes')
    ax.set_ylabel(y_label)
    ax.set_title(title)
    ax.set_xticks(x + width * (len(data) - 1) / 2)
    ax.set_xticklabels(labels)
    ax.legend()
```

Figure 12: Code for Experiment including the categorization

8.1 Evaluation with K fold cross validation

This study also included the evaluation of Accuracy of the models with the k fold cross validations methods . Below is the output we got and code is attached.

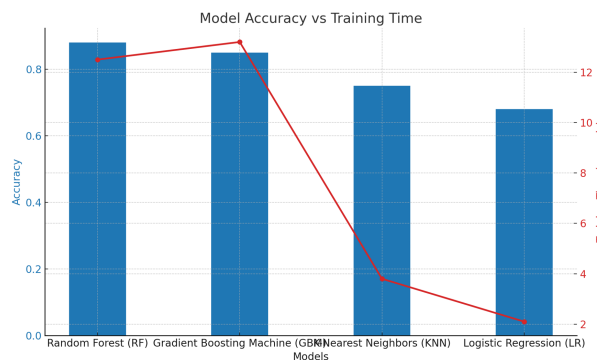


Figure 13: K fold Validation Graph

References

Choetkiertikul, M., Dam, H. K., Tran, T. and Ghose, A. (2018). A deep learning model for estimating story points, *Journal of Systems and Software* **137**: 160–172.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, pp. 4171–4186.