Configuration Manual

MSc Research Project
December 2024 Winter Submission

Aman Lenka
Student ID: X23176351

School of Computing
National College of Ireland

Supervisor: Professor Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Aman Lenka		
Student ID:	X23176351		
Programme:	MSc Data Analytics	**Year:**	2024
Module:	MSc Research Project December 2024 Winter Submission		
Lecturer:	Noel Cosgrave		
Submission Due Date:	12th December 2024		
Project Title:	Predictive Analytics for Reducing Patient Readmission Rates in the Healthcare Sector		
Word Count:	**970 Page Count: 6**		

Signature: AMAN LENKA

Date: 10th December 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

Author: Aman Lenka
Student ID: X23176351

Project Title: Predictive Analytics for Reducing Patient Readmission Rates in the Healthcare Sector

This configuration manual provides instructions to set up, install and run the artifacts associated with the "Grid-Copy 1" (Keras Tuner Grid Search with CV) and "Bayesian-Copy" (Optuna Bayesian Optimization with CV) scripts. These scripts implement advanced hyperparameter tuning with 5-fold cross-validation and train predictive models for hospital readmission classification.

1 System Requirements

Hardware Requirements:

- CPU: AMD Ryzen 7 7840HS or equivalent
- GPU: NVIDIA RTX 4060 (optional for faster training)
- RAM: 16–32 GB recommended
- Storage: ~50 GB free for data, models, and logs

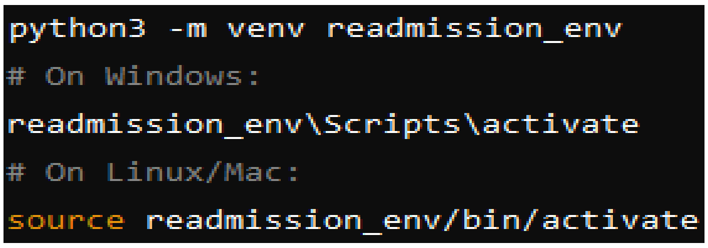
Software Requirements:

- Operating System: Windows 10/11 (64-bit) or Ubuntu 20.04 LTS
- Python 3.10
- (Optional) CUDA & cuDNN if using GPU acceleration.

2 Environment Setup

Step-by-Step Setup (using venv and pip):

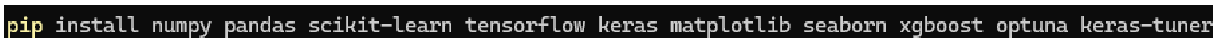
1. Ensure Python 3.10 is installed.
2. Create and activate a virtual environment:



```
python3 -m venv readmission_env
# On Windows:
readmission_env\Scripts\activate
# On Linux/Mac:
source readmission_env/bin/activate
```

Fig.1: Command prompt for setting up a virtual environment.

3. Install the required libraries:



```
pip install numpy pandas scikit-learn tensorflow keras matplotlib seaborn xgboost optuna keras-tuner
```

Fig. 2: PowerShell commands for installing required libraries.

Note:

- tensorflow should be installed in a version compatible with your GPU and CUDA drivers if you want GPU acceleration.
- keras-tuner is required for Grid Search-based tuning.
- optuna is used for Bayesian Optimization.

If you prefer conda, create and activate an environment similarly, then install packages via conda install or pip.

3 Data Preparation

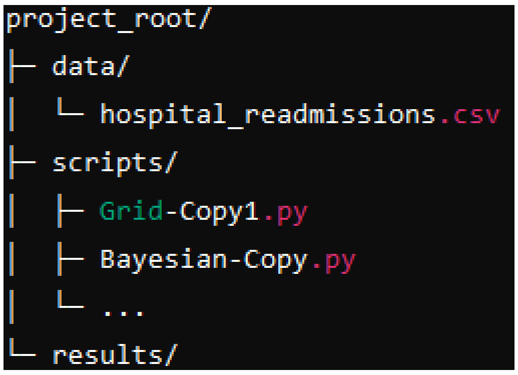
Data Source:

The dataset file hospital_readmissions.csv should be obtained from the UCI ML Repository/Kaggle.

Place it in a data/ directory: (for my case it was):
C:/Users/amanl/OneDrive/Research in Computing/hospital_readmissions.csv

This will be different for you. Use the file path you have instead of the one mentioned above. Usually, this will be where you download the files.

Alternatively, you can make this data directory and place it there:

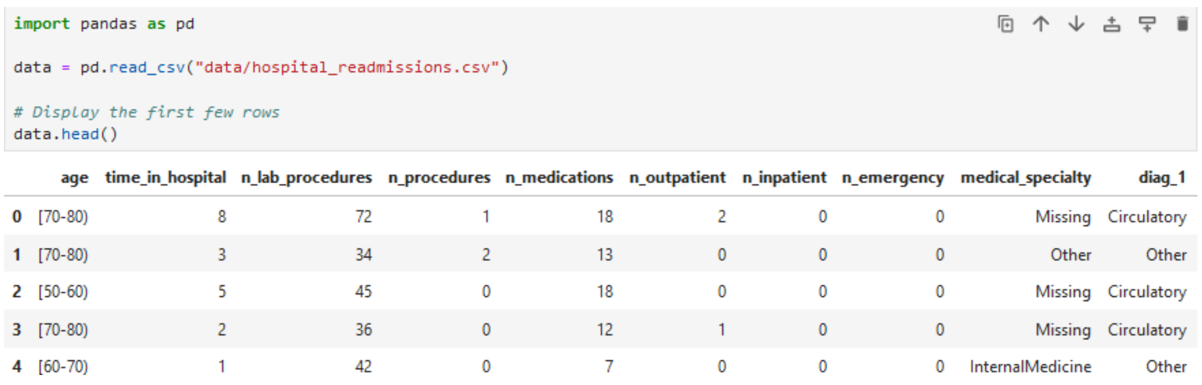


```
project_root/
├ data/
│   └ hospital_readmissions.csv
├ scripts/
│   ├ Grid-Copy1.py
│   ├ Bayesian-Copy.py
│   └ ...
└ results/
```

Fig. 3: Directory structure for organizing project data.

Verifying the Data:

```
import pandas as pd
df = pd.read_csv("data/hospital_readmissions.csv")
print(df.head())
```



```python
import pandas as pd

data = pd.read_csv("data/hospital_readmissions.csv")

# Display the first few rows
data.head()
```

	age	time_in_hospital	n_lab_procedures	n_procedures	n_medications	n_outpatient	n_inpatient	n_emergency	medical_specialty	diag_1
0	[70-80)	8	72	1	18	2	0	0	Missing	Circulatory
1	[70-80)	3	34	2	13	0	0	0	Other	Other
2	[50-60)	5	45	0	18	0	0	0	Missing	Circulatory
3	[70-80)	2	36	0	12	1	0	0	Missing	Circulatory
4	[60-70)	1	42	0	7	0	0	0	InternalMedicine	Other

Fig. 4: Code snippet from Jupyter Environment about the data

4 Running the Grid-Copy1 Script (Keras Tuner with CV)

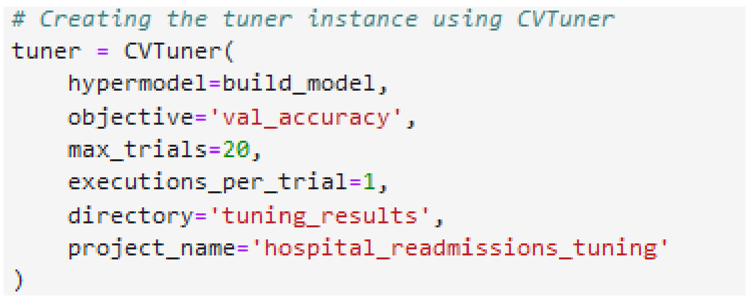
Script Name: Grid-Copy1.py

This script:

- Performs hyperparameter tuning using Keras Tuner with a custom CVTuner class implementing 5-fold cross-validation.
- Searches for optimal neural network architecture and training hyperparameters.
- Prints out the best hyperparameters and trains a final model with them**.**

Steps to Run:
python scripts/Grid-Copy1.py

Expected Output:
- Console output showing the tuning trials and the best hyperparameters found.
- Training logs for the final model with chosen hyperparameters.
- Performance metrics, confusion matrices, and plotted figures (accuracy, loss, ROC curves).



```python
# Creating the tuner instance using CVTuner
tuner = CVTuner(
    hypermodel=build_model,
    objective='val_accuracy',
    max_trials=20,
    executions_per_trial=1,
    directory='tuning_results',
    project_name='hospital_readmissions_tuning'
)
```

Fig. 5: Creating a Tuner instance using CV.

5 Running the Bayesian-Copy Script (Optuna Bayesian Optimization with CV)

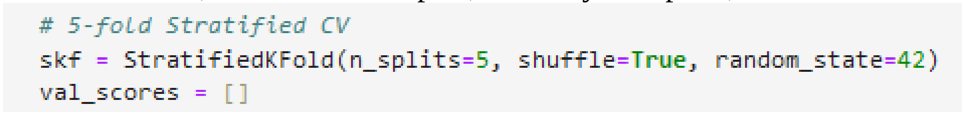
Script Name: Bayesian-Copy.py

This script:

- Uses Optuna to perform Bayesian Optimization of hyperparameters.
- Incorporates StratifiedKFold 5-fold CV directly inside the objective function.
- After optimization, it trains the final model with the best hyperparameters, evaluates it on a test set, and plots metrics.

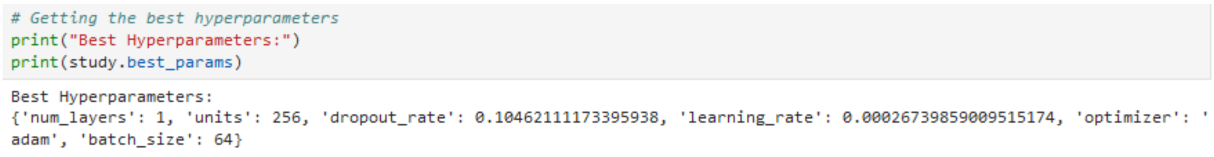
Steps to Run:
python scripts/Bayesian-Copy.py

Expected Output:
- Console output with best hyperparameters selected by Optuna.
- Final model performance metrics printed in the terminal.
- Confusion matrix, classification report, accuracy/loss plots, and ROC curve displayed.



```
# 5-fold Stratified CV
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
val_scores = []
```

Fig. 6: Using stratified CV for Bayesian Optimization



```
# Getting the best hyperparameters
print("Best Hyperparameters:")
print(study.best_params)
```

```
Best Hyperparameters:
{'num_layers': 1, 'units': 256, 'dropout_rate': 0.10462111173395938, 'learning_rate': 0.00026739859009515174, 'optimizer': '
adam', 'batch_size': 64}
```

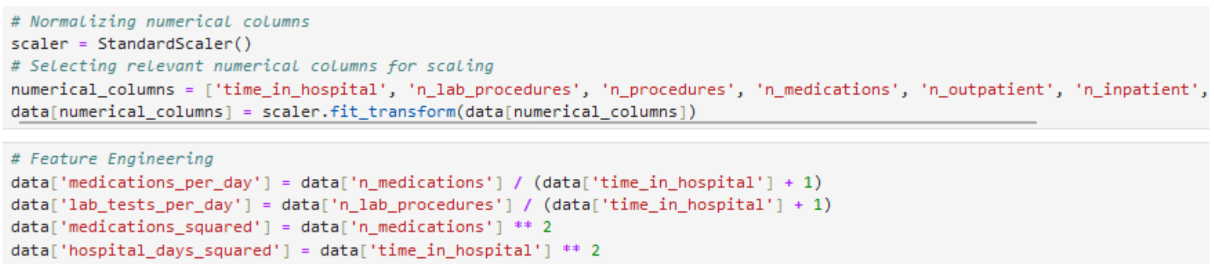
Fig. 7: Best Parameters

Note: The script writes tuning results to a 'tuning_results/hospital_readmissions_tuning' directory (specified in the code). Ensure you have 'write' permissions in that directory, or adjust the path as needed.

6 Pipeline Implementation Details

Both scripts follow a similar pipeline:

1. **Data Preprocessing:**
   - Missing values handling, encoding categorical features, normalization of continuous features.
   - Feature engineering (e.g., medications_per_day, lab_tests_per_day).



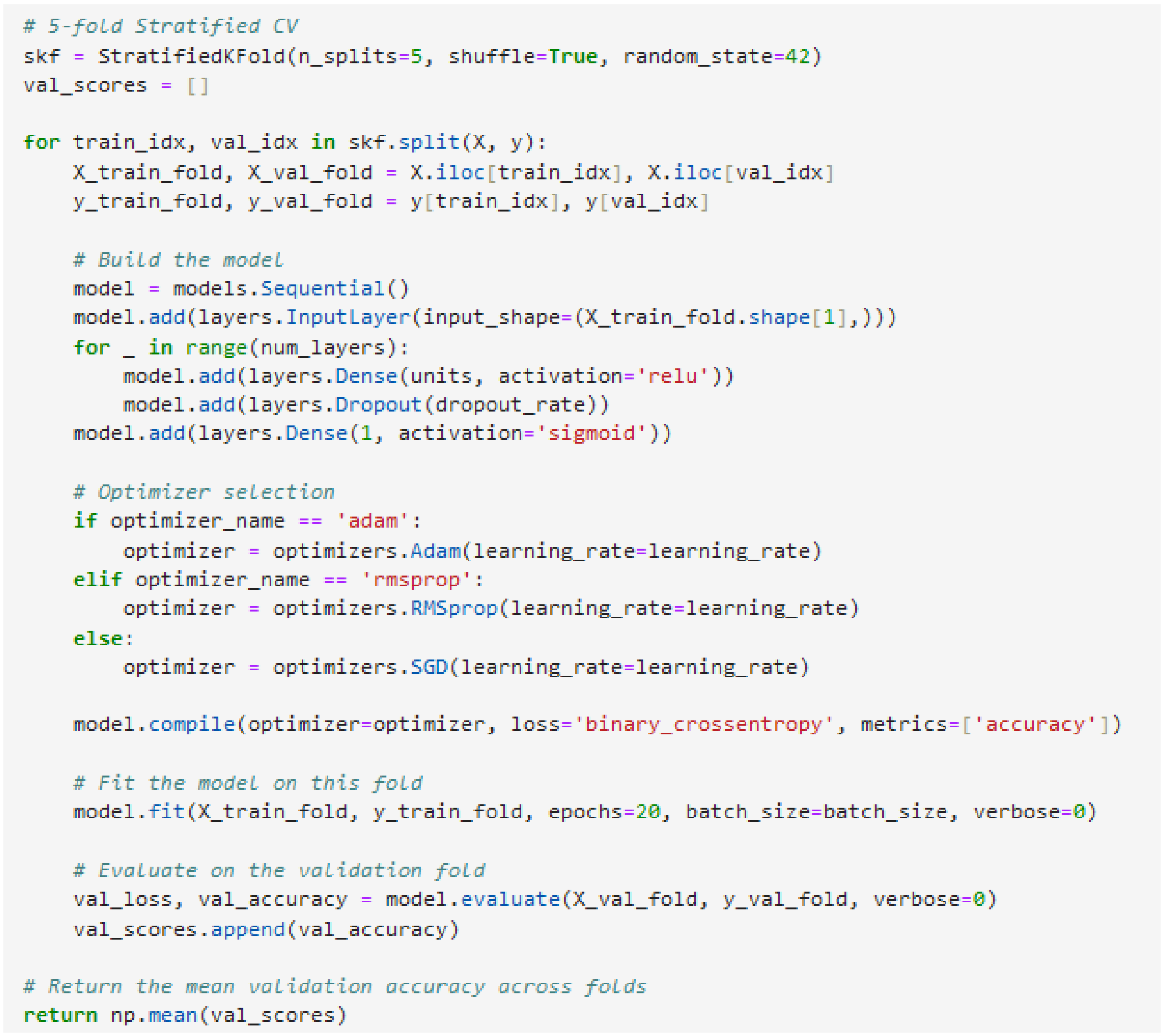
```
# Normalizing numerical columns
scaler = StandardScaler()
# Selecting relevant numerical columns for scaling
numerical_columns = ['time_in_hospital', 'n_lab_procedures', 'n_procedures', 'n_medications', 'n_outpatient', 'n_inpatient',
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])
```

```
# Feature Engineering
data['medications_per_day'] = data['n_medications'] / (data['time_in_hospital'] + 1)
data['lab_tests_per_day'] = data['n_lab_procedures'] / (data['time_in_hospital'] + 1)
data['medications_squared'] = data['n_medications'] ** 2
data['hospital_days_squared'] = data['time_in_hospital'] ** 2
```

Fig. 8: Preprocessing snippet

2. **Model Definition and Hyperparameter Space:**
   - For Grid-Copy1: build_model(hp) function defines the search space for Keras Tuner.
   - For Bayesian-Copy: The objective(trial) function defines search space for Optuna.

3. **Cross-Validation Integration:**
    - Grid-Copy1: Custom CVTuner class overriding run_trial.
    - Bayesian-Copy: StratifiedKFold inside the objective function.



```python
# 5-fold Stratified CV
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
val_scores = []

for train_idx, val_idx in skf.split(X, y):
    X_train_fold, X_val_fold = X.iloc[train_idx], X.iloc[val_idx]
    y_train_fold, y_val_fold = y[train_idx], y[val_idx]

    # Build the model
    model = models.Sequential()
    model.add(layers.InputLayer(input_shape=(X_train_fold.shape[1],)))
    for _ in range(num_layers):
        model.add(layers.Dense(units, activation='relu'))
        model.add(layers.Dropout(dropout_rate))
    model.add(layers.Dense(1, activation='sigmoid'))

    # Optimizer selection
    if optimizer_name == 'adam':
        optimizer = optimizers.Adam(learning_rate=learning_rate)
    elif optimizer_name == 'rmsprop':
        optimizer = optimizers.RMSprop(learning_rate=learning_rate)
    else:
        optimizer = optimizers.SGD(learning_rate=learning_rate)

    model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

    # Fit the model on this fold
    model.fit(X_train_fold, y_train_fold, epochs=20, batch_size=batch_size, verbose=0)

    # Evaluate on the validation fold
    val_loss, val_accuracy = model.evaluate(X_val_fold, y_val_fold, verbose=0)
    val_scores.append(val_accuracy)

# Return the mean validation accuracy across folds
return np.mean(val_scores)
```

Fig. 9: Stratified Kfold usage

4. **Final Model Evaluation:**
    - After hyperparameter tuning, both scripts train a final model on the designated training set and evaluate on a separate test set.
    - Metrics (accuracy, precision, recall, F1-score, AUC) and plots (confusion matrix, ROC, training history) are generated.

7 Tools and Frameworks

- **Programming Language:** Python 3.10
- **Core Libraries:**
    - numpy, pandas, scikit-learn for data handling and preprocessing.
    - tensorflow, keras, keras-tuner for building and tuning neural networks.
    - xgboost for baseline model comparison.
    - optuna for Bayesian optimization.

    - matplotlib, seaborn for plotting.
- **Additional Requirements:**
    - Ensure pip is up to date: pip install --upgrade pip.

8 Troubleshooting and Common Issue

- **ModuleNotFoundError:** Check if the environment is activated and all dependencies are installed.
- **GPU issues:** Make sure CUDA and cuDNN are properly installed if using GPU. Otherwise, install tensorflow-cpu.
- **Memory Errors:** Reduce batch size, number of layers, or run on a machine with more RAM.
- **File Not Found for Dataset:** Update the data/hospital_readmissions.csv path in the scripts if you placed the dataset elsewhere.

9 Version Control and Updates

If you are going to use a version control system (e.g., Git), commit code changes regularly. If updates are made to these scripts, describe them in a CHANGELOG.md file. For reproducing exact experiments, note down the commit hash that corresponds to final reported results.

References

Conda *Conda documentation*. Available at: https://docs.conda.io/ (Accessed: 08 December 2024).
CUDA *CUDA Toolkit Documentation*. NVIDIA Corporation. Available at: https://docs.nvidia.com/cuda/ (Accessed: 10 December 2024).
cuDNN *cuDNN Installation Guide*. NVIDIA Corporation. Available at: https://docs.nvidia.com/deeplearning/cudnn/ (Accessed: 10 December 2024).
Keras *Keras Documentation*. Available at: https://keras.io/ (Accessed: 10 December 2024).
Keras Tuner *Keras Tuner Documentation*. Available at: https://keras.io/keras_tuner/ (Accessed: 10 December 2024).
NCI Library *Referencing Guide: Harvard style*. National College of Ireland Library. Available at: https://libguides.ncirl.ie/referencing (Accessed: 10 December 2024).
Optuna *Optuna Documentation*. Available at: https://optuna.org/ (Accessed: 10 December 2024).
Python Software Foundation *Python 3.10 Downloads*. Available at: https://www.python.org/downloads/ (Accessed: 10 December 2024).
scikit-learn *scikit-learn Documentation*. Available at: https://scikit-learn.org/stable/ (Accessed: 10 December 2024).
TensorFlow *TensorFlow Documentation*. Available at: https://www.tensorflow.org/ (Accessed: 10 December 2024).
XGBoost *XGBoost Documentation*. Available at: https://xgboost.readthedocs.io/ (Accessed: 10 December 2024).