National College of Ireland

Efficient and Scalable Retail Demand Forecasting with Sparse Attention Mechanisms

MSc Research Project
Data Analytics

Akhilesh Thangellapally
Student ID: 23283793

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Akhilesh Thangellapally
Student ID:	23283793
Programme:	MSc in Data Analytics **Year:** 2025-2026
Module:	Research Practicum
Lecturer:	Bharat Agarwal
Submission Due Date:	27/01/2026
Project Title:	Efficient and Scalable Retail Demand Forecasting with Sparse Attention Mechanisms
Word Count:	**810** **Page Count: 5**

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:	Akhilesh Thangellapally
Date:	27/01/2026

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):	

Efficient and Scalable Retail Demand Forecasting with Sparse Attention Mechanisms

Akhilesh Thangellapally
Student ID: 23283793

1. Introduction

This section outlines the general objective of the project: the development of a scalable and high precision retail demand forecasting system. It describes the difficulties of retail data: high dimensionality, mixed feature types, and time complexity, and points out why the conventional models fail. It proposes a solution known as the Sparse Attention model, which can cut down the amount of computation, but also increase accuracy in forecasting.

2. System Specifications

Hardware Requirements

- CPU: Multi-core Intel/AMD processor
- GPU: NVIDIA CUDA-enabled GPU (e.g., Tesla T4, V100, RTX-series)
- RAM: Minimum 8 GB; recommended 16 GB for faster training
- Storage: Minimum 10 GB free space for data, models, and logs

Software Requirements

- Operating System: Ubuntu 20.04/22.04, Windows 10/11, or Google Colab environment
- Python Version: Python 3.10+
- IDE/Environment: Jupiter Notebook / Google Colab / VS Code
- CUDA Version: CUDA 11.x or above (for GPU acceleration)
- cuDNN Version: cuDNN 8.x (for TensorFlow GPU compatibility)

Core Frameworks

- TensorFlow 2.15 (GPU enabled)
- Scikit-learn (classical ML models & preprocessing)
- LightGBM, Cat Boost (tree-based ensemble models)

3. Required Libraries

The application is based on core scientific libraries, machine learning systems, and deep learning toolkits. NumPy and Pandas are used to calculate matrices and manipulate data. Exploratory data visualization can be facilitated with Matplotlib and Seaborn, which allow identifying the patterns of distribution, the seasonality, and variations at the category level. Scikit-learn has pre-processing, scaling, train-test splitting, classical regression models and hyperparameter search. Cat Boost and LightGBM provide tree-based ensemble architectures which are optimized. To build deep learning and transformer-based models, the Sparse Attention layers, as well as to build deep learning and transformers models, TensorFlow and Keras are used. Lastly, utility modules like warnings, random and OS are used to aid in environment stability and reproducibility.

Installation Commands:

pip install numpy pandas matplotlib seaborn scikit-learn
pip install catboost lightgbm
pip install tensorflow

4. Data description

The data includes retail transaction-level data that includes past information on product characteristics, store numbers, sales data, price changes, inventory quantities, and external environmental characteristics. The variables in each entry are: Category, Region, Weather Condition, Seasonality, Inventory Level, Units Sold, Units ordered, Price, Discount, Competitor Pricing and Holiday/Promotion. Demand Forecast is the target variable, which is a numeric demand prediction of each product-store combination. The Date variable which is initially in the form of a string allows the extraction of temporal trends whereby we can extract monthly and seasonal trends.

5. Models Implemented

This part describes the development of the models that were utilized in the research. It begins with classical ML models as control groups, proceeds to deep learning models such as MLP and GRU to nonlinearly model relations, and ends with the Sparse Attention Transformer. The section explains the reasons why each model was selected and its contribution towards the overall assessment.

6. Code Snippets

a) Checking for missing values:

```
df.isnull().sum()
```

	0
Date	0
Store ID	0
Product ID	0
Category	0
Region	0
Inventory Level	0
Units Sold	0
Units Ordered	0
Demand Forecast	0
Price	0
Discount	0
Weather Condition	0
Holiday/Promotion	0
Competitor Pricing	0

b) Describing the dataset:

```
df.describe()
```

	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount	Holiday/Promotion	Competitor Pricing
count	73100.000000	73100.000000	73100.000000	73100.000000	73100.000000	73100.000000	73100.000000	73100.000000
mean	274.469877	136.464870	110.004473	141.494720	55.135108	10.009508	0.497305	55.146077
std	129.949514	108.919406	52.277448	109.254076	26.021945	7.083746	0.499996	26.191408
min	50.000000	0.000000	20.000000	-9.990000	10.000000	0.000000	0.000000	5.030000
25%	162.000000	49.000000	65.000000	53.670000	32.650000	5.000000	0.000000	32.680000
50%	273.000000	107.000000	110.000000	113.015000	55.050000	10.000000	0.000000	55.010000
75%	387.000000	203.000000	155.000000	208.052500	77.860000	15.000000	1.000000	77.820000
max	500.000000	499.000000	200.000000	518.550000	100.000000	20.000000	1.000000	104.940000

c) Train test split:

```
X = df.drop(columns=['Demand Forecast', 'Date', 'Price_Bin','Store ID','Product ID','Competitor Pricing'])
y = df['Demand Forecast']
```

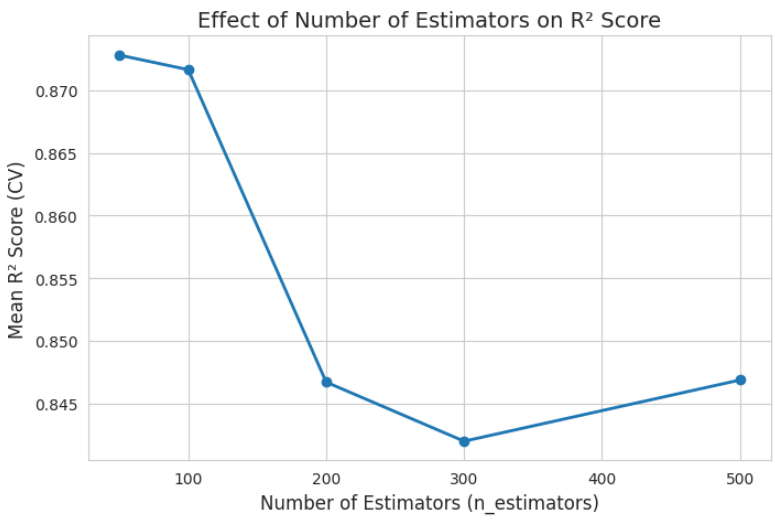
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False, random_state=SEED
)

X_train.shape, X_test.shape
```

```
((58480, 11), (14620, 11))
```

d) Hyperparameter plot for random forest:



e) Main sparse attension model:

```
class SparseAttention(layers.Layer):
    def __init__(self, num_heads, key_dim, window=2, dropout=0.1):
        super().__init__()
        self.num_heads = num_heads
        self.key_dim = key_dim
        self.window = window
        self.dropout = layers.Dropout(dropout)
        self.mha = layers.MultiHeadAttention(num_heads=num_heads, key_dim=key_dim)

    def build_sparse_mask(self, L):
        i = tf.range(L)[:, None]
        j = tf.range(L)[None, :]
        dist = tf.abs(i - j)
        mask = tf.where(dist <= self.window, 1.0, 0.0)
        mask = tf.expand_dims(tf.expand_dims(mask, 0), 0)
        return mask

    def call(self, x, training=False):
        L = tf.shape(x)[1]
        mask = self.build_sparse_mask(L)
        attn_output = self.mha(x, x, attention_mask=mask)
        attn_output = self.dropout(attn_output, training=training)
        return attn_output
```

7. Performance Figure Description:

In table 1, it can be seen that there is a distinct division between the classical ML, the deep learning models, and the Sparse Attention architecture. Although machine learning models comprise a middle ground and deep learning networks enhance accuracy by learning nonlinear representations, Sparse Attention Model is always better than the other ones. The fact that its error values are very low and its R2 is almost perfect means that it is able to capture both the local and long-term dependencies by removing inconsequential features through the use of sparsity constraints. The visual representation of the comparison plotted highlights the high level of accuracy that is offered through the sparse attention mechanism.

Table 1: Model Evaluation Metrics

Model	MAE	MSE	RMSE	R^2 Score
Random Forest	0.0609	0.0064	0.0802	0.8476
Cat Boost	0.0783	0.0094	0.0971	0.7766
Light GBM	0.0665	0.0072	0.0849	0.8292
MLP	0.0551	0.0045	0.0670	0.8936
RNN (GRU)	0.0553	0.0048	0.0690	0.8870
Sparse Attention Model	**0.0176**	**0.0005**	**0.0215**	**0.9890**

8. Conclusion:

The architecture given in this report provides a fully reproducible forecasting pipeline starting with dataset pre-processing and moving to the high-level attention-based modelling. Classical ML models offer strong baselines, deep learning models offer improved accuracy and the Sparse Attention Transformer offers state-of-the-art performance with rich embedding structures, and with efficient local attention windows. The fact that it has very high R^2 and low error statistics proves that it is superior in this forecasting pipeline.

The workflow can be extended and can be modified to multi-store forecasting, hierarchical modelling, probabilistic predictions or can be combined with online retail decision platforms. The Sparse Attention architecture specifically offers an efficient and multi-purpose base of large-scale operational deployment in retail settings

References:

1. **https://seaborn.pydata.org/**
2. **https://www.tensorflow.org/api_docs**
3. **https://keras.io/api/**
4. **https://pandas.pydata.org/docs/**
5. **https://numpy.org/doc/**
6. **https://www.tensorflow.org/api_docs/python/tf/keras/layers/MultiHeadAttention**