COMP4332 Group 4 Project 3 Rating Prediction Report

Introduction

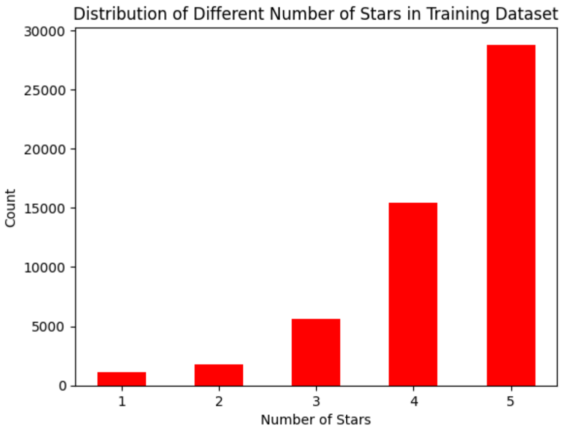
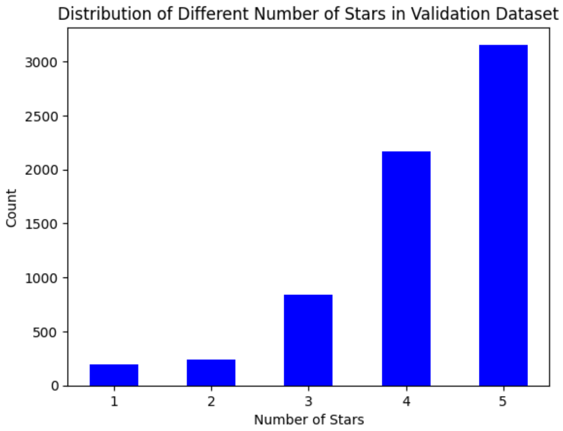
In this project, our objective is to predict users' ratings on various items using a recommendation system. The accuracy of our predictions is evaluated using the Root Mean Squared Error (RMSE) metric. To achieve this, we implement a Wide & Deep Learning (WDL) model, which combines the strengths of both linear models and deep learning models. This hybrid approach leverages the memorization capabilities of linear models and the generalization capabilities of deep neural networks to improve prediction accuracy.

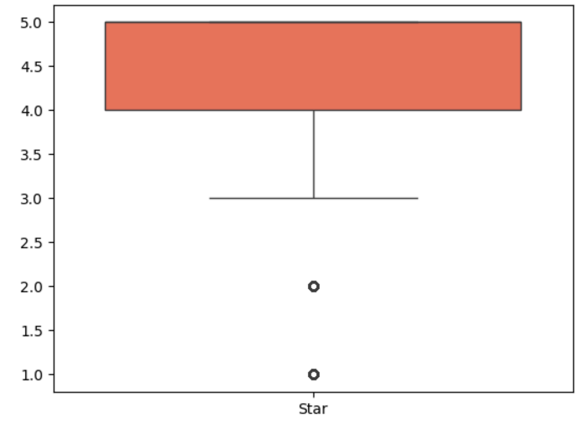
Data Exploration

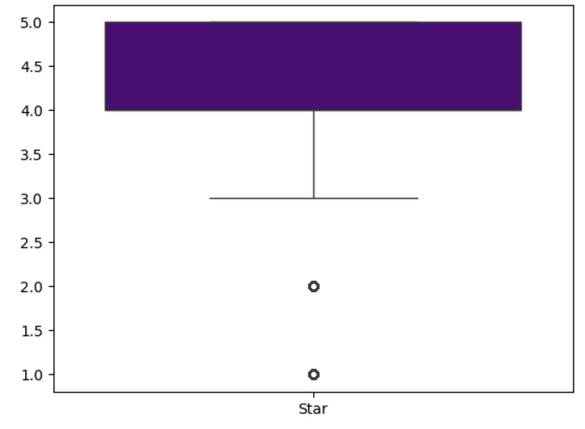
We have done some data exploration on both the training dataset and the validation dataset. In particular, we have explored (i) the distribution of different stars, (ii) the statistics of ProductID against total number of stars received, and (iii) the statistics of ReviewerID against total number of stars received for both training dataset and testing dataset respectively.

(i) the distribution of different stars

Below are the bar charts and boxplots for the distribution of different stars in training dataset and validation dataset respectively:

Training Dataset	Validation Dataset
	



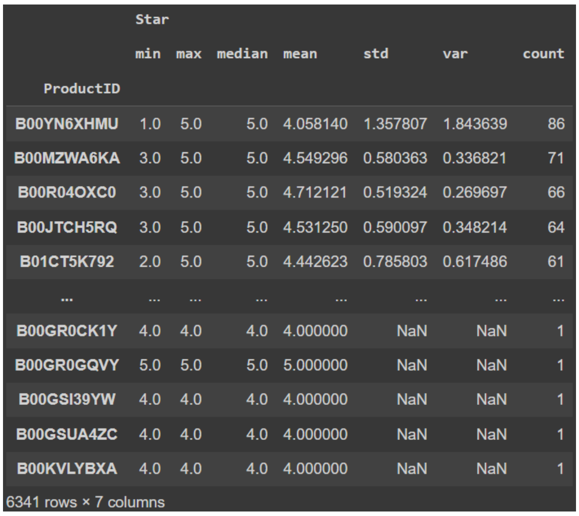


The key observation here is that for both the training dataset and the testing dataset, the overall ratings are mostly 4 stars or 5 stars, with only few of them being equal or below 3 stars. Although there is an uneven distribution of the good ratings, neutral ratings and poor ratings, but given the hint that the validation set and testing set are randomly split from the data within the same time period as stated on Canvas, this kind of data skewness should not affect the testing performance by too much.

<u>(ii) the statistics of ProductID against total number of stars received</u>

Below are the statistics summary and the bar charts for the ProductID against total number of stars received in training dataset and validation dataset respectively:

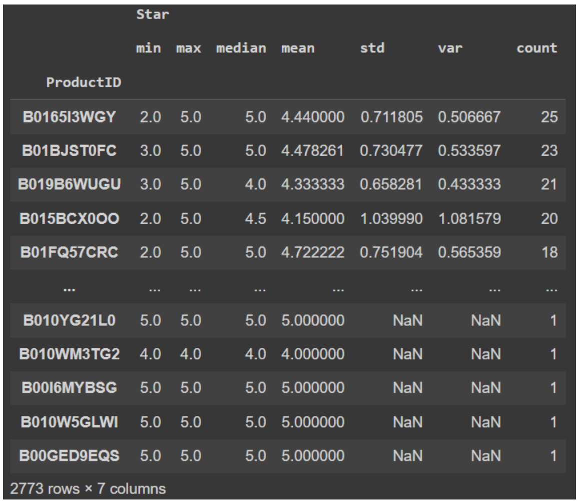
Training Dataset



	Star						
	min	max	median	mean	std	var	count
ProductID							
B00YN6XHMU	1.0	5.0	5.0	4.058140	1.357807	1.843639	86
B00MZWA6KA	3.0	5.0	5.0	4.549296	0.580363	0.336821	71
B00R04OXC0	3.0	5.0	5.0	4.712121	0.519324	0.269697	66
B00JTCH5RQ	3.0	5.0	5.0	4.531250	0.590097	0.348214	64
B01CT5K792	2.0	5.0	5.0	4.442623	0.785803	0.617486	61
...	...	...	...	...	...	...	...
B00GR0CK1Y	4.0	4.0	4.0	4.000000	NaN	NaN	1
B00GR0GQVY	5.0	5.0	5.0	5.000000	NaN	NaN	1
B00GSI39YW	4.0	4.0	4.0	4.000000	NaN	NaN	1
B00GSUA4ZC	4.0	4.0	4.0	4.000000	NaN	NaN	1
B00KVLYBXA	4.0	4.0	4.0	4.000000	NaN	NaN	1

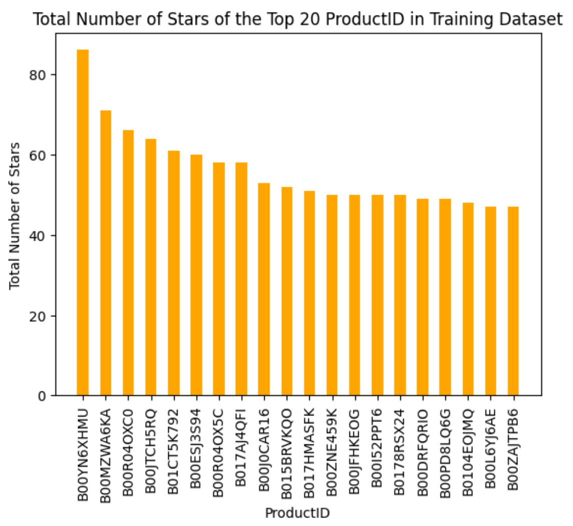
6341 rows × 7 columns

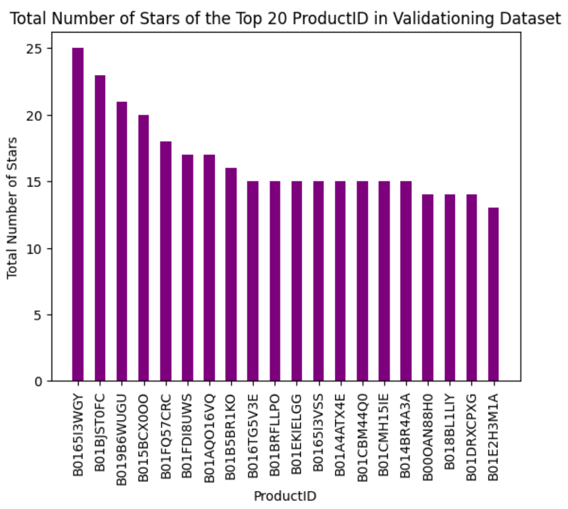
Validation Dataset



	Star						
	min	max	median	mean	std	var	count
ProductID							
B0165I3WGY	2.0	5.0	5.0	4.440000	0.711805	0.506667	25
B01BJST0FC	3.0	5.0	5.0	4.478261	0.730477	0.533597	23
B019B6WUGU	3.0	5.0	4.0	4.333333	0.658281	0.433333	21
B015BCX0OO	2.0	5.0	4.5	4.150000	1.039990	1.081579	20
B01FQ57CRC	2.0	5.0	5.0	4.722222	0.751904	0.565359	18
...	...	...	...	...	...	...	...
B010YG21L0	5.0	5.0	5.0	5.000000	NaN	NaN	1
B010WM3TG2	4.0	4.0	4.0	4.000000	NaN	NaN	1
B00I6MYBSG	5.0	5.0	5.0	5.000000	NaN	NaN	1
B010W5GLWI	5.0	5.0	5.0	5.000000	NaN	NaN	1
B00GED9EQS	5.0	5.0	5.0	5.000000	NaN	NaN	1

2773 rows × 7 columns



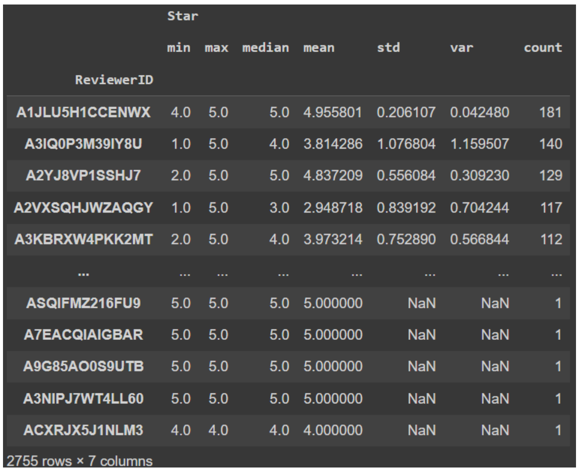


The key observation is that the total number of stars obtained by the Top 20 ProductID in both training dataset and testing dataset seems to have the shadow of “long tail” effect and the 80-20 rule (80% of the ratings comes from the top 20% of the products), which proves the idea of long tail effect exists in recommendation system.

(iii) the statistics of ReviewerID against total number of stars received

Below are the statistics summary and the bar charts for the RevirerID against total number of stars received in training dataset and validation dataset respectively:

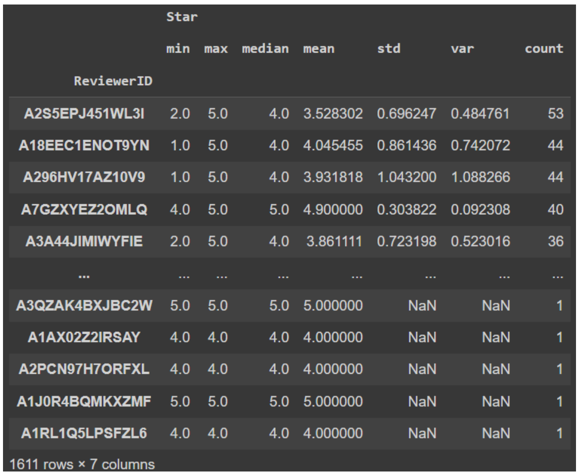
Training Dataset



	Star						
	min	max	median	mean	std	var	count
ReviewerID							
A1JLU5H1CCENWX	4.0	5.0	5.0	4.955801	0.206107	0.042480	181
A3IQ0P3M39IY8U	1.0	5.0	4.0	3.814286	1.076804	1.159507	140
A2YJ8VP1SSHJ7	2.0	5.0	5.0	4.837209	0.556084	0.309230	129
A2VXSQHJWZAQGY	1.0	5.0	3.0	2.948718	0.839192	0.704244	117
A3KBRXW4PKK2MT	2.0	5.0	4.0	3.973214	0.752890	0.566844	112
...	...	...	...	...	...	...	...
ASQIFMZ216FU9	5.0	5.0	5.0	5.000000	NaN	NaN	1
A7EACQIAIGBAR	5.0	5.0	5.0	5.000000	NaN	NaN	1
A9G85AO0S9UTB	5.0	5.0	5.0	5.000000	NaN	NaN	1
A3NIPJ7WT4LL60	5.0	5.0	5.0	5.000000	NaN	NaN	1
ACXRJX5J1NLM3	4.0	4.0	4.0	4.000000	NaN	NaN	1

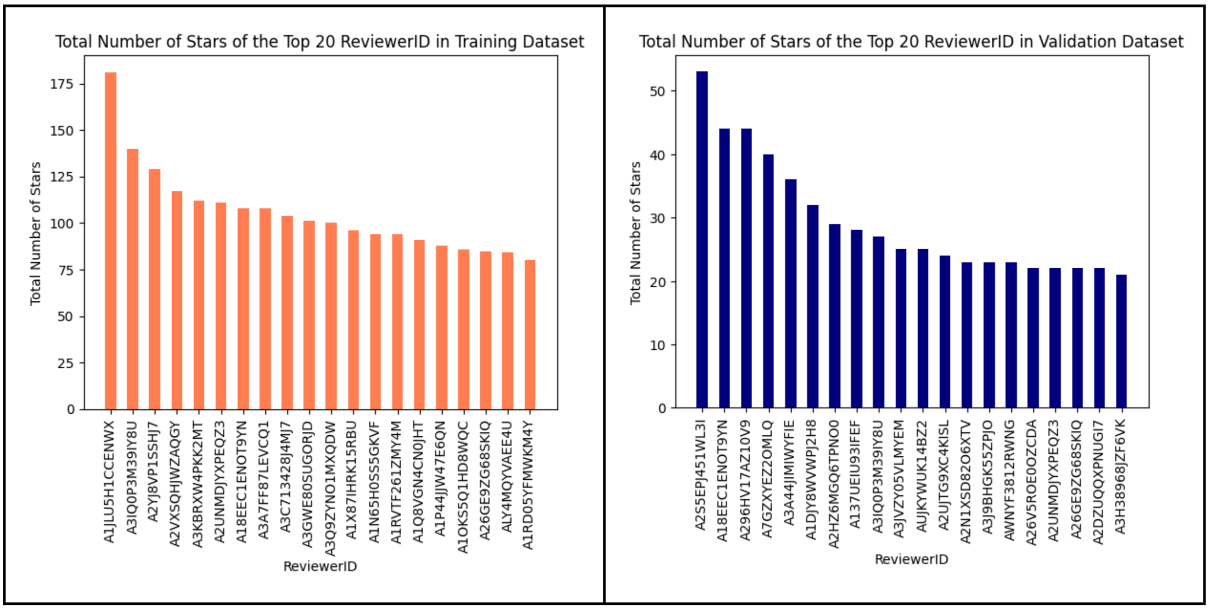
2755 rows × 7 columns

Validation Dataset



	Star						
	min	max	median	mean	std	var	count
ReviewerID							
A2S5EPJ451WL3I	2.0	5.0	4.0	3.528302	0.696247	0.484761	53
A18EEC1ENOT9YN	1.0	5.0	4.0	4.045455	0.861436	0.742072	44
A296HV17AZ10V9	1.0	5.0	4.0	3.931818	1.043200	1.088266	44
A7GZXYEZ2OMLQ	4.0	5.0	5.0	4.900000	0.303822	0.092308	40
A3A44JIMIWYFIE	2.0	5.0	4.0	3.861111	0.723198	0.523016	36
...	...	...	...	...	...	...	...
A3QZAK4BXJBC2W	5.0	5.0	5.0	5.000000	NaN	NaN	1
A1AX02Z2IRSAY	4.0	4.0	4.0	4.000000	NaN	NaN	1
A2PCN97H7ORFXL	4.0	4.0	4.0	4.000000	NaN	NaN	1
A1J0R4BQMKXZMF	5.0	5.0	5.0	5.000000	NaN	NaN	1
A1RL1Q5LPSFZL6	4.0	4.0	4.0	4.000000	NaN	NaN	1

1611 rows × 7 columns



The key observation is similar to that in (ii). The total number of stars obtained by the Top 20 ReviewerID in both training dataset and testing dataset seems to have the shadow of “long tail” effect and the 80-20 rule (80% of the ratings comes from the top 20% of the reviewers), which again proves the idea of long tail effect exists in recommendation system.

Data Processing

The dataset provided includes user ratings, user reviews, and extra product information. Specifically, it consists of several files: review.csv with 52,697 ratings and reviews, validation.csv containing 6,596 (reviewer, product, rating) triples, prediction.csv with 6,597 (reviewer, product) pairs where the "Star" column is set to 0.0, and product.json detailing 6,734 products, though not all products are guaranteed to be included. We begin by loading these datasets using the pandas library, followed by preprocessing steps to extract relevant features. This involves merging datasets based on common keys, handling missing values, standardizing data formats, encoding categorical features, and normalizing continuous features. The processed data is then split into training and validation sets. The training set is used to train our model, while the validation set helps in tuning the model and evaluating its performance.

Model

We utilize the Wide & Deep Learning (WDL) model, which merges two components: the wide component and the deep component. The wide component is a generalized linear model that processes raw input features and cross-product transformations of categorical features, enabling it to learn co-occurrence patterns effectively. This enhances the model's memorization ability. The deep component, on the other hand, is a Feed-forward Neural Network (FNN) that handles both continuous and categorical features. It includes embedding layers for categorical features, multiple dense layers with ReLU activation functions, and dropout layers for regularization to prevent overfitting.

The model architecture involves input layers for continuous and categorical features, embedding layers for categorical features, and dense layers for the deep component interspersed with dropout layers. The wide component is a linear model that processes wide inputs. The outputs of the wide and deep components are concatenated and passed through a final dense layer to produce the predicted rating. The model is compiled with the Adam optimizer and Mean Squared Error (MSE) loss function.

To ensure reproducibility, we set random seeds for Python's random module, NumPy, and TensorFlow. The model is then trained using the training data, and its performance is evaluated on the validation data. The combination of the wide and deep components allows the model to effectively capture both memorization and generalization aspects, leading to improved prediction accuracy.

Results

The performance of the Wide & Deep Learning (WDL) model was evaluated on both the training and validation datasets. We used the Root Mean Squared Error (RMSE) as the primary metric to assess the accuracy of our predictions. The results are as follows:

- Training RMSE: 0.7186
- Validation RMSE: 0.8235

The RMSE values indicate that while the model fits the training data reasonably well, it shows a slight increase in error on the validation dataset, suggesting some degree of overfitting.

Classification Metrics

In addition to RMSE, we also evaluated the model using precision, recall, and F1-score metrics. These metrics provide a more detailed view of the model's performance across different rating categories. Below are the results:

Rating	**Precision**	**Recall**	**F1-Score**	**Support**
1.0	0.00	0.00	0.00	166
2.0	0.25	0.13	0.17	234
3.0	0.31	0.31	0.31	800

4.0	0.46	0.64	0.53	2076
5.0	0.76	0.61	0.68	2888
6.0	0.00	0.00	0.00	0

- **Overall Accuracy:** 0.55; **Macro Avg Precision:** 0.30; **Macro Avg Recall:** 0.28; **Macro Avg F1-Score:** 0.28; **Weighted Avg Precision:** 0.56; **Weighted Avg Recall:** 0.55; **Weighted Avg F1-Score:** 0.54

Figures

The figures in the appendix provide a detailed analysis of the Wide & Deep Learning (WDL) model's performance and architecture.

- Figure 1 shows the training and validation accuracy over 20 epochs. Both training and validation accuracies exhibit minimal improvement and plateau early, indicating potential underfitting.
- Figure 2 displays the training and validation loss. The training loss decreases initially but stabilizes thereafter, while the validation loss remains steady, further supporting the underfitting hypothesis.
- Figure 3 presents the confusion matrix on validation dataset. It reveals that the model performs well for some rating categories but struggles significantly with others, particularly categories 1 and 3.
- Figure 4 expressed the overall architecture of our Wide & Deep Model.

These findings suggest that while the WDL model has potential, it requires further optimization and possibly more complex techniques to improve its predictive performance across all rating categories.

Conclusion

The performance analysis of the Wide & Deep Learning (WDL) model highlights both its strengths and areas for improvement. While the training and validation RMSE values indicate that the model can reasonably fit the training data, the slight increase in validation error suggests a degree of overfitting.

The classification metrics reveal that the model performs well for higher rating categories but struggles significantly with lower ratings, as evidenced by precision, recall, and F1-scores. Additionally, the figures illustrate that the model's training and validation accuracy quickly plateau, further indicating potential underfitting.

These findings suggest that while the WDL model shows promise, further optimization, including more complex modeling techniques and fine-tuning, is necessary to enhance its predictive performance across all rating categories. We recommend the exploration of regularization methods and more sophisticated model architectures to address these issues and improve overall model accuracy.

Appendix

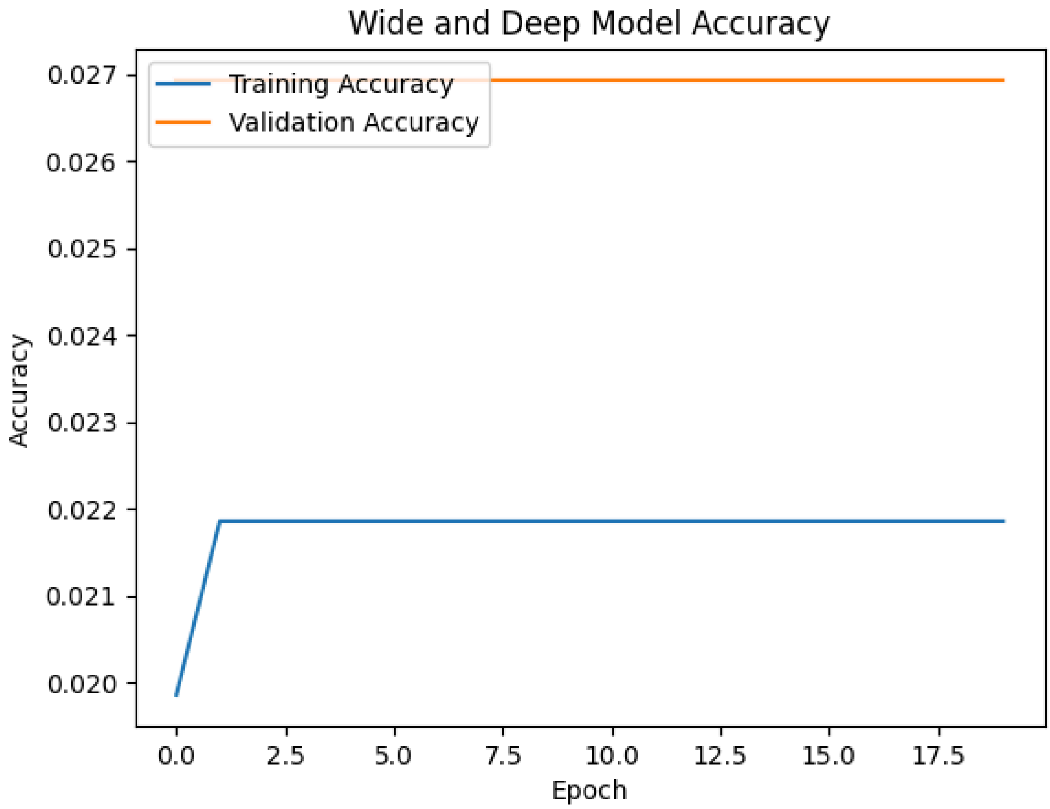


Figure1: Wide and Deep Model Accuracy

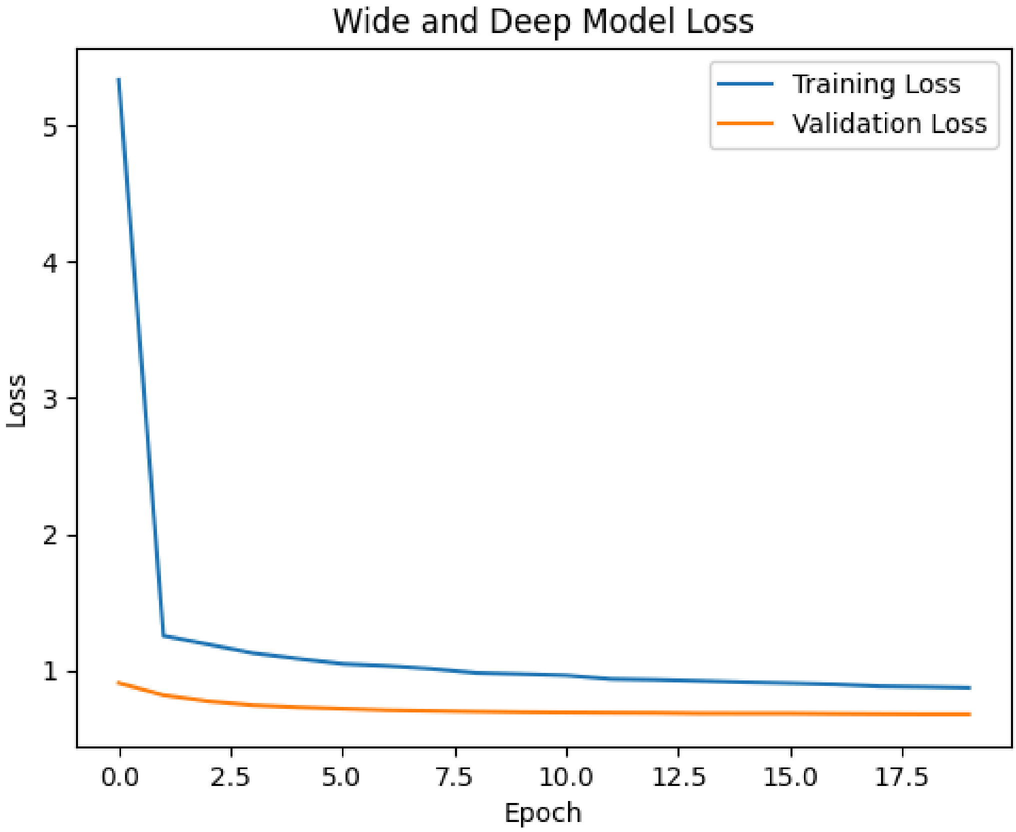
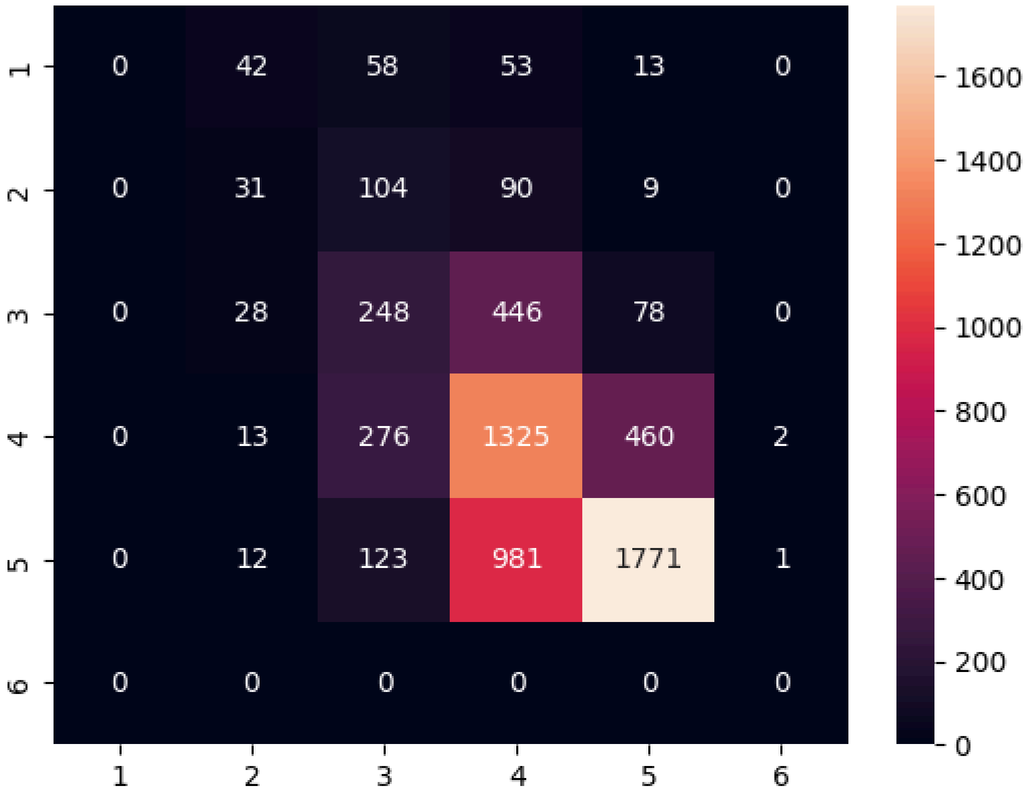


Figure2: Wide and Deep Model Loss



	1	2	3	4	5	6
1	0	42	58	53	13	0
2	0	31	104	90	9	0
3	0	28	248	446	78	0
4	0	13	276	1325	460	2
5	0	12	123	981	1771	1
6	0	0	0	0	0	0

Figure3: Visualization of Confusion Matrix on validation dataset

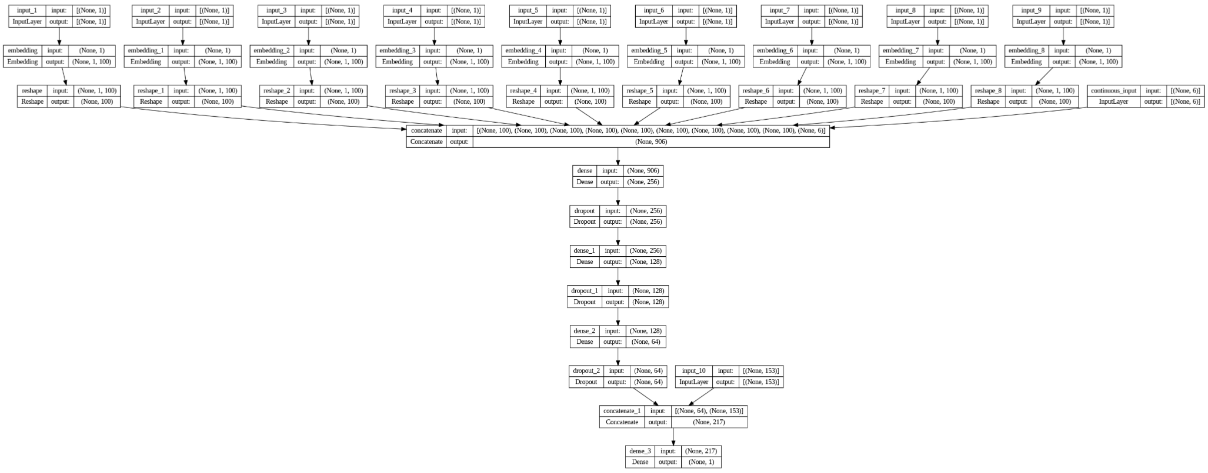

input_1 InputLayer input: [(None, 1)] output: [(None, 1)]
input_2 InputLayer input: [(None, 1)] output: [(None, 1)]
input_3 InputLayer input: [(None, 1)] output: [(None, 1)]
input_4 InputLayer input: [(None, 1)] output: [(None, 1)]
input_5 InputLayer input: [(None, 1)] output: [(None, 1)]
input_6 InputLayer input: [(None, 1)] output: [(None, 1)]
input_7 InputLayer input: [(None, 1)] output: [(None, 1)]
input_8 InputLayer input: [(None, 1)] output: [(None, 1)]
input_9 InputLayer input: [(None, 1)] output: [(None, 1)]
embedding Embedding input: (None, 1) output: (None, 1, 100)
embedding_1 Embedding input: (None, 1) output: (None, 1, 100)
embedding_2 Embedding input: (None, 1) output: (None, 1, 100)
embedding_3 Embedding input: (None, 1) output: (None, 1, 100)
embedding_4 Embedding input: (None, 1) output: (None, 1, 100)
embedding_5 Embedding input: (None, 1) output: (None, 1, 100)
embedding_6 Embedding input: (None, 1) output: (None, 1, 100)
embedding_7 Embedding input: (None, 1) output: (None, 1, 100)
embedding_8 Embedding input: (None, 1) output: (None, 1, 100)
reshape Reshape input: (None, 1, 100) output: (None, 100)
reshape_1 Reshape input: (None, 1, 100) output: (None, 100)
reshape_2 Reshape input: (None, 1, 100) output: (None, 100)
reshape_3 Reshape input: (None, 1, 100) output: (None, 100)
reshape_4 Reshape input: (None, 1, 100) output: (None, 100)
reshape_5 Reshape input: (None, 1, 100) output: (None, 100)
reshape_6 Reshape input: (None, 1, 100) output: (None, 100)
reshape_7 Reshape input: (None, 1, 100) output: (None, 100)
reshape_8 Reshape input: (None, 1, 100) output: (None, 100)
continuous_input InputLayer input: [(None, 6)] output: [(None, 6)]
concatenate Concatenate input: [(None, 100), (None, 100), (None, 100), (None, 100), (None, 100), (None, 100), (None, 100), (None, 100), (None, 100), (None, 6)] output: (None, 906)
dense Dense input: (None, 906) output: (None, 256)
dropout Dropout input: (None, 256) output: (None, 256)
dense_1 Dense input: (None, 256) output: (None, 128)
dropout_1 Dropout input: (None, 128) output: (None, 128)
dense_2 Dense input: (None, 128) output: (None, 64)
dropout_2 Dropout input: (None, 64) output: (None, 64)
input_10 InputLayer input: [(None, 153)] output: [(None, 153)]
concatenate_1 Concatenate input: [(None, 64), (None, 153)] output: (None, 217)
dense_3 Dense input: (None, 217) output: (None, 1)

Figure4: Visualization of the overall architecture of our Wide & Deep Model