CREDIT SCORE CLASSIFICATION

Presented By:

Andre Herndon, Ayush Choudhary, Priyanshi Verma, Thor Abbasi, Tathagata Saha

The problem justification

Traditional credit scoring models like FICO Score and VantageScore have been traditionally used in determining credit worthiness.

Despite their widespread use, these models often fail to reflect the nuance in consumer financial behavior and habits.

Recent findings by Experian and OliverWyman reveal that approximately 91 million Americans are excluded from accessing standard rate credit due to the limitations of traditional scoring algorithms.These models tend to overlook nuanced aspects of financial profiles, impacting significant portions of the population.

Related Work

STRENGTHS

- Comprehensive review of classical and modern classification techniques for credit scoring.
- Rigorous empirical comparison across multiple datasets provides insights into strengths and weaknesses.
- Highlighted issues like dealing with imbalanced classes and feature selection importance.

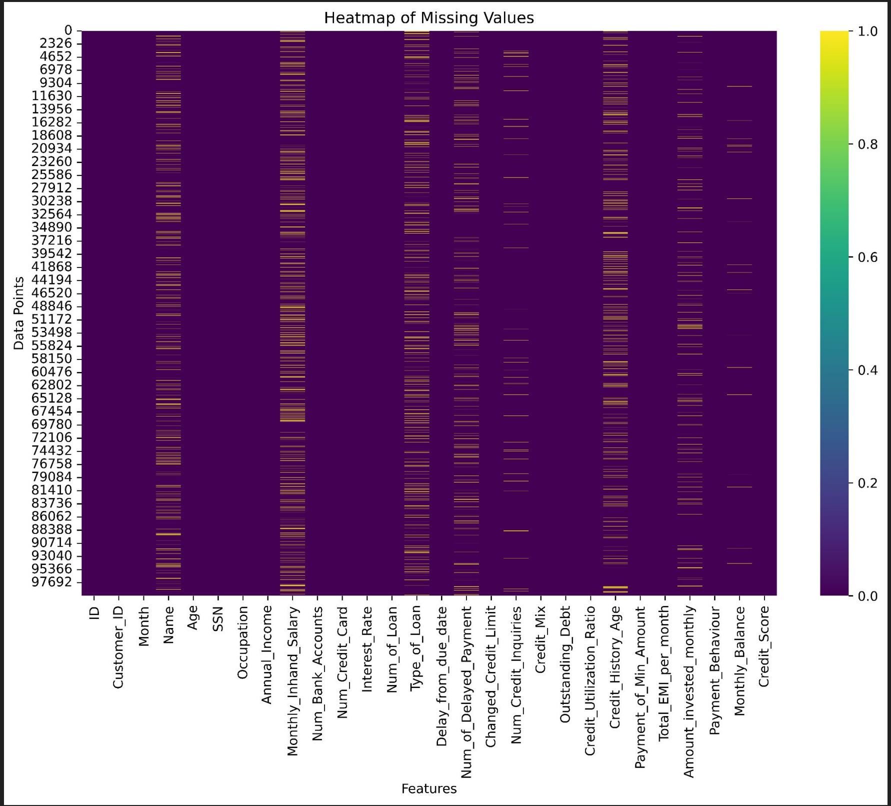
WEAKNESS

- Did not cover very recent state-of-the-art models like boosted trees or deep learning.
- Only considered discrimination ability, did not evaluate other criteria like interpretability.
- Did not propose any novel methods, only reviewed and tested existing approaches.

Relevant dataset

- We used a publicly available dataset containing information commonly used to determine the credit score of an individual.

- The data set contains:
  - 150,000 unique records
  - 27 unique attributes
  - 90,124 records missing at least one attribute
  - Numerous records contained inaccurate data



Method choice and implementation

Attention Mechanism:Transformers leverage attention mechanisms to focus on relevant features, crucial for imbalanced datasets where certain features are more indicative of creditworthiness.

Sequence Modeling:Well-suited for sequential data like monthly financial transactions, transformers capture dependencies between past and current events, aiding in predicting credit scores based on historical behavior.

Non-local Relationships:Transformers excel at capturing long-range dependencies and non-local relationships, beneficial for identifying subtle patterns between various financial indicators contributing to creditworthiness.

Scale and Complexity:With strong performance on extensive datasets, transformers handle complexity better than traditional architectures, accommodating large feature sets common in credit scoring tasks.

Potential for Transfer Learning:Leveraging pre-trained models like BERT or GPT allows for transfer learning, fine-tuning models on credit scoring tasks to benefit from learned generalizations.

Future Expansion:Transformer-based architectures offer flexibility for evolving project requirements and dataset characteristics, accommodating changes or additional features/data sources over time.

Experiment Design

Variance Inflation Factor (VIF) and SMOTE:

- VIF is applied to identify and mitigate multicollinearity among features. Features with high VIF scores are removed to improve model interpretability and stability.
- Synthetic Minority Over-sampling Technique (SMOTE) addresses class imbalance by generating synthetic samples for minority class instances, balancing the class distribution during model training.

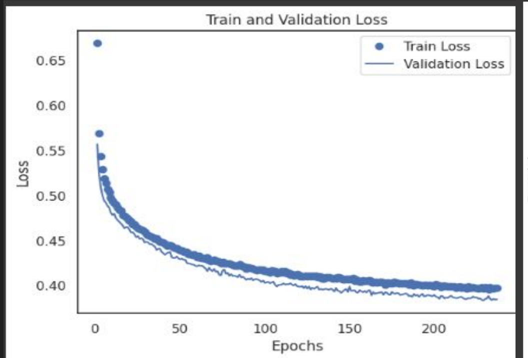
Model Selection:

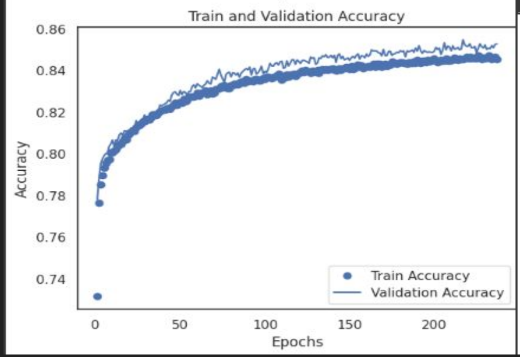
- TensorFlow and Keras are utilized to implement neural network models.
- Two primary architectures are considered: a transformer-based model and a traditional feed-forward neural network (FNN) model.

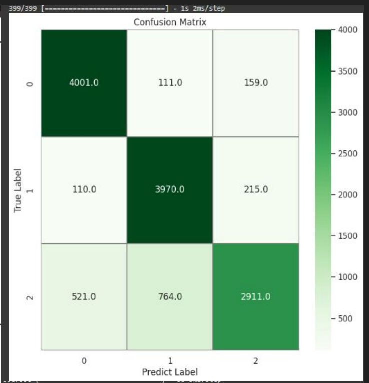
Evaluation Metrics:

- Model performance is assessed using multiple evaluation metrics, including accuracy, precision, recall, F1-score, and Area under the ROC curve (AUC-ROC).
- Model interpretability is also considered to evaluate the impact of VIF on feature selection and SMOTE on class imbalance handling.

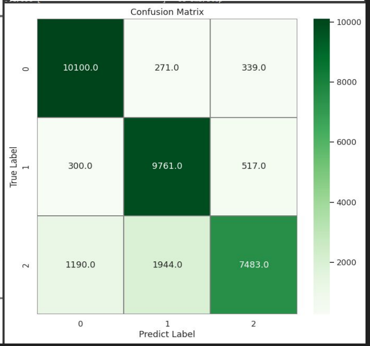
Results

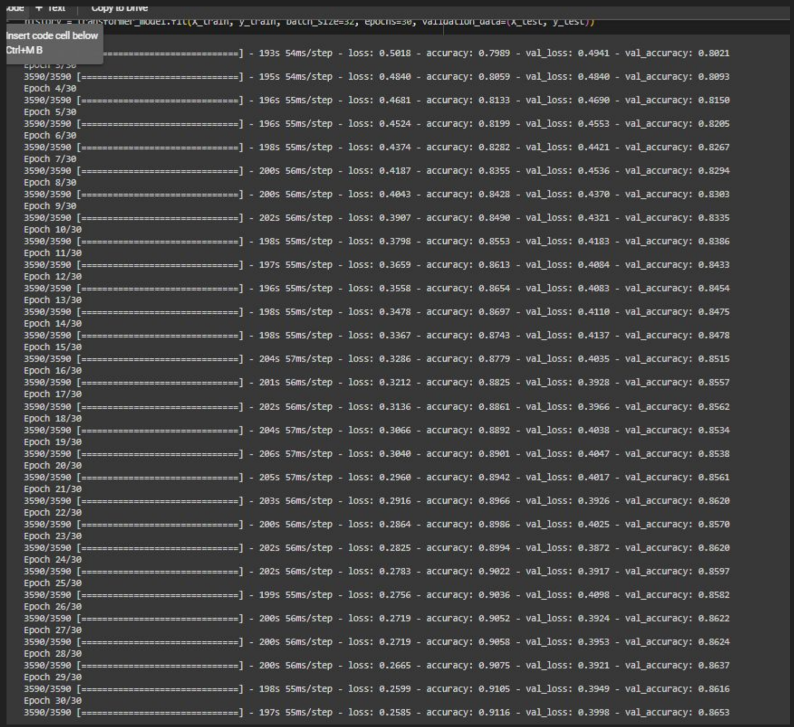






```
399/399 [==============================] - 1s 2ms/step
```





```
history = transformer_model.fit(X_train, y_train, batch_size=32, epochs=30, validation_data=(X_test, y_test))

==========================] - 193s 54ms/step - loss: 0.5018 - accuracy: 0.7989 - val_loss: 0.4941 - val_accuracy: 0.8021
Epoch 3/30
3590/3590 [==============================] - 195s 54ms/step - loss: 0.4840 - accuracy: 0.8059 - val_loss: 0.4840 - val_accuracy: 0.8093
Epoch 4/30
3590/3590 [==============================] - 196s 55ms/step - loss: 0.4681 - accuracy: 0.8133 - val_loss: 0.4690 - val_accuracy: 0.8150
Epoch 5/30
3590/3590 [==============================] - 196s 55ms/step - loss: 0.4524 - accuracy: 0.8199 - val_loss: 0.4553 - val_accuracy: 0.8205
Epoch 6/30
3590/3590 [==============================] - 198s 55ms/step - loss: 0.4374 - accuracy: 0.8282 - val_loss: 0.4421 - val_accuracy: 0.8267
Epoch 7/30
3590/3590 [==============================] - 200s 56ms/step - loss: 0.4187 - accuracy: 0.8355 - val_loss: 0.4536 - val_accuracy: 0.8294
Epoch 8/30
3590/3590 [==============================] - 200s 56ms/step - loss: 0.4043 - accuracy: 0.8428 - val_loss: 0.4370 - val_accuracy: 0.8303
Epoch 9/30
3590/3590 [==============================] - 202s 56ms/step - loss: 0.3907 - accuracy: 0.8490 - val_loss: 0.4321 - val_accuracy: 0.8335
Epoch 10/30
3590/3590 [==============================] - 198s 55ms/step - loss: 0.3798 - accuracy: 0.8553 - val_loss: 0.4183 - val_accuracy: 0.8386
Epoch 11/30
3590/3590 [==============================] - 197s 55ms/step - loss: 0.3659 - accuracy: 0.8613 - val_loss: 0.4084 - val_accuracy: 0.8433
Epoch 12/30
3590/3590 [==============================] - 196s 55ms/step - loss: 0.3558 - accuracy: 0.8654 - val_loss: 0.4083 - val_accuracy: 0.8454
Epoch 13/30
3590/3590 [==============================] - 198s 55ms/step - loss: 0.3478 - accuracy: 0.8697 - val_loss: 0.4110 - val_accuracy: 0.8475
Epoch 14/30
3590/3590 [==============================] - 198s 55ms/step - loss: 0.3367 - accuracy: 0.8743 - val_loss: 0.4137 - val_accuracy: 0.8478
Epoch 15/30
3590/3590 [==============================] - 204s 57ms/step - loss: 0.3286 - accuracy: 0.8779 - val_loss: 0.4035 - val_accuracy: 0.8515
Epoch 16/30
3590/3590 [==============================] - 201s 56ms/step - loss: 0.3212 - accuracy: 0.8825 - val_loss: 0.3928 - val_accuracy: 0.8557
Epoch 17/30
3590/3590 [==============================] - 202s 56ms/step - loss: 0.3136 - accuracy: 0.8861 - val_loss: 0.3966 - val_accuracy: 0.8562
Epoch 18/30
3590/3590 [==============================] - 204s 57ms/step - loss: 0.3066 - accuracy: 0.8892 - val_loss: 0.4038 - val_accuracy: 0.8534
Epoch 19/30
3590/3590 [==============================] - 206s 57ms/step - loss: 0.3040 - accuracy: 0.8901 - val_loss: 0.4047 - val_accuracy: 0.8538
Epoch 20/30
3590/3590 [==============================] - 205s 57ms/step - loss: 0.2960 - accuracy: 0.8942 - val_loss: 0.4017 - val_accuracy: 0.8561
Epoch 21/30
3590/3590 [==============================] - 203s 56ms/step - loss: 0.2916 - accuracy: 0.8966 - val_loss: 0.3926 - val_accuracy: 0.8620
Epoch 22/30
3590/3590 [==============================] - 200s 56ms/step - loss: 0.2864 - accuracy: 0.8986 - val_loss: 0.4025 - val_accuracy: 0.8570
Epoch 23/30
3590/3590 [==============================] - 202s 56ms/step - loss: 0.2825 - accuracy: 0.8994 - val_loss: 0.3872 - val_accuracy: 0.8620
Epoch 24/30
3590/3590 [==============================] - 202s 56ms/step - loss: 0.2783 - accuracy: 0.9022 - val_loss: 0.3917 - val_accuracy: 0.8597
Epoch 25/30
3590/3590 [==============================] - 199s 55ms/step - loss: 0.2756 - accuracy: 0.9036 - val_loss: 0.4098 - val_accuracy: 0.8582
Epoch 26/30
3590/3590 [==============================] - 200s 56ms/step - loss: 0.2719 - accuracy: 0.9052 - val_loss: 0.3924 - val_accuracy: 0.8622
Epoch 27/30
3590/3590 [==============================] - 200s 56ms/step - loss: 0.2719 - accuracy: 0.9058 - val_loss: 0.3953 - val_accuracy: 0.8624
Epoch 28/30
3590/3590 [==============================] - 200s 56ms/step - loss: 0.2665 - accuracy: 0.9075 - val_loss: 0.3921 - val_accuracy: 0.8637
Epoch 29/30
3590/3590 [==============================] - 198s 55ms/step - loss: 0.2599 - accuracy: 0.9105 - val_loss: 0.3949 - val_accuracy: 0.8616
Epoch 30/30
3590/3590 [==============================] - 197s 55ms/step - loss: 0.2585 - accuracy: 0.9116 - val_loss: 0.3998 - val_accuracy: 0.8653
```

Conclusion and Future work

Takeaways:

- We were able to successfully implement a neural network for the purposes of credit scoring.
- While the model performed well there is room for improvement.

Next Steps:

- **Enhance Model Accuracy:** Implement additional feature engineering techniques and parameter optimization.
- **Improve Interpretability:** Make the models decisions transparent to help users understand the logic used in decisioning.
- **Model Stability:** Ensure that the model is just as effective so that it is usable in real-world scenarios.