# CONVOLVE 3.0 SOLUTION REPORT

## EXPLORATORY DATA ANALYSIS

---

### Data Loading and Initial Inspection

The project commenced with loading the dataset into a Pandas DataFrame for initial exploration and analysis. The dataset consisted of **96,806 rows and 1,216 columns**, which indicated a substantial number of features.

The first step involved addressing two primary concerns:

1. **Feature Reduction**: The large number of columns necessitated a systematic approach to identify and retain only the most relevant features for the prediction task.
2. **Handling Missing Values**: Several columns contained missing (NaN) values, which required appropriate cleaning strategies before training the model.

### Handling Missing Values

To address the issue of missing values, we performed a detailed analysis by using the `isnull().sum()` function to compute the count of missing values across all columns. The results were saved to a CSV file for thorough inspection and review.

### Dropping Columns with Significant Missing Values

To address missing values in the dataset, we identified and removed columns with **over 30,000 missing values.** This threshold represents approximately 30% of the total data points, ensuring that columns with significant data deficiencies were excluded from further analysis.

Given the high proportion of missing data in these columns, retaining them would have added noise and complexity to the model without contributing meaningful information. Consequently, these columns were dropped from the dataset to reduce dimensionality.

### Dropping Rows with Significant Missing Values

Upon analysis, we discovered that several columns had exactly **25,231** missing values.Further investigation revealed that the missing values were concentrated in the same rows across these columns.

Given the extent of missing data and the uniformity of missing rows across affected columns, we decided to drop the rows with missing values. This decision ensured the dataset's consistency while retaining the majority of the data points for further analysis.

## Dropping Rows with Clustered Missing Values

During iterative checks for null values following each removal, we discovered several attributes with the same number of missing values. The count of missing values in these attributes ranged from **199 to 1,100**, which represents a small fraction of the remaining dataset containing **71,575** data points after the previous step. Given their minimal impact on the dataset, we removed these rows systematically in subsequent steps.
This left us with  69341 rows.

## Dropping Columns further and Filling the rest of missing values

We dropped columns with still more than 20000 missing values and filled the rest with the median values of corresponding columns.

With all missing values successfully addressed and removed, the dataset was prepared for the next phase of the process: dimensionality reduction.

## Feature Reduction

To address the high dimensionality of the dataset, we implemented a systematic approach to reduce the number of features while retaining the most relevant information for our predictive model. This involved the following steps:

1. **Variance Thresholding**:
   The first step in the dimensionality reduction process was variance thresholding. An analysis of the variance across all columns revealed **a range from 0 to 2.35e+14**, with a median value of 0.07. Considering the large number of features and the substantial variance in columns exceeding this threshold, we opted to remove features with a **variance below 0.07**. This step ensured that low-variance features, which contribute minimal information, were excluded from further analysis.This left us with **603 features**.

2. **Removing Highly Correlated Features**:

   **A) Pairwise Correlation:**

   Excluding the target column, we computed the pairwise correlation coefficients between all remaining features. Features with a correlation coefficient **greater than 0.9** were identified as highly correlated and subsequently removed. This step reduced the number of **features to 414**, ensuring that redundant information was minimized.

**B) Correlation With Target:**

Next, we calculated the correlation coefficient of each remaining feature with the target variable. The threshold for this step was determined based on the performance of the model trained on features selected at different thresholds.

- At a threshold of **0.05, 26** features were retained.
- At **0.03**, approximately **70** features remained.
- At **0.02,** the number of features increased to **around 100**.
- At **0.01, 189** features were preserved.

After analyzing the model's performance across these thresholds, a correlation threshold of **0.01** was found to yield the optimal results. This value was selected to balance feature selection and model performance.

These steps were crucial in improving the computational efficiency and predictive performance of the model by minimizing redundancy and noise in the dataset.

## MODEL DEVELOPMENT AND TRAINING

---

To develop an effective predictive model, we adopted a structured approach to training and tuning. Here, we outline the steps taken, along with the key numerical results obtained at each stage.

### Data Preparation

We split the dataset into training and testing sets, allocating 67% of the data for training and 33% for testing. To address the disparity in feature scales, we standardized the data using the `StandardScaler`.

The significant class imbalance in the dataset led us to calculate a class weight of approximately **67.623** for the positive class to counteract the imbalance.

---

### Sequential Neural Network

We designed a baseline neural network with the following structure:

- **Input layer**: Corresponding to the number of features (dimensionality of the dataset).
- **Hidden layers**: Three dense layers with 256, 128, and 64 neurons, respectively, using **ReLU** activations and dropout rates of 0.3 and 0.2 for regularization.
- **Output layer**: A single neuron with a **sigmoid activation** function for binary classification.

**Hyperparameters**:

- Optimizer: Adam with a learning rate of 0.001.

- Loss Function: Binary Cross-Entropy.
- Batch Size: 64.
- Epochs: 30 (with early stopping applied after 5 epochs of no improvement in validation loss).

**Results of the Baseline Neural Network**:

- Accuracy: **76.60%**
- Precision: **4.44%**
- Recall: **69.32%**
- F1 Score: **8.35%**
- Confusion Matrix:

  [ [ 17285  5246 ]

  [ 108  244 ] ]

While the accuracy was relatively high, the low precision and F1 score highlighted the model's difficulty in identifying the minority class.

---

## ResNet-Inspired Neural Network

To improve performance, we implemented a ResNet-inspired neural network with residual connections to enhance gradient flow and prevent vanishing gradients.

**Architecture**:

- **Initial Dense Layer**: 128 neurons with ReLU activation, batch normalization, and a dropout rate of 0.5.
- **Residual Blocks**: Three blocks, each with two dense layers (128 neurons) followed by batch normalization and dropout, with residual connections added.
- **Output Layer**: A single neuron with a sigmoid activation.

**Hyperparameters**:

- Optimizer: Adam with a learning rate of 0.001.
- Loss Function: Binary Cross-Entropy.
- Batch Size: 32.
- Epochs: 100 (with early stopping applied after 10 epochs of no improvement in validation loss).

**Results of the ResNet-Inspired Neural Network**:

- Accuracy: **77.94%**
- Precision: **4.72%**
- Recall: **69.60%**
- F1 Score: **8.85%**
- Confusion Matrix:

  [ [ 17589  4942 ]

  [ 107  245 ] ]

Compared to the baseline, the ResNet-inspired model improved precision, recall, and F1 score, particularly in identifying the minority class (positive samples).

---

**Hyperparameter Tuning**

During the parameter tuning phase of the neural network model, we evaluated multiple loss functions, including **`dice_loss` and a custom precision loss function**. However, none of these alternatives significantly improved the model's performance compared to **binary cross-entropy, which we ultimately selected as the loss function.**

Additionally, we experimented with increasing the number of layers in the network architecture. However, this adjustment did not yield any notable improvement in performance, leading us to retain the original architecture.

We fine-tuned the ResNet model by adjusting:

- Number of neurons in hidden layers: 128.
- Dropout rates: 0.5 for the input layer and 0.3 for hidden layers.
- Learning rate: 0.001.

This tuning led to more robust performance.

---

## Use of Mean Absolute Error (MAE) in Model Evaluation

In addition to standard classification metrics such as accuracy, precision, recall, and F1 score, we utilized **Mean Absolute Error (MAE)** to gain an alternative perspective on the model's performance. This choice was guided by the evaluation criteria, which specified the assessment of absolute distance from the true probability. MAE provided a straightforward and effective measure for comparing predicted probabilities to their true values.

The primary purpose of employing MAE was to measure the average magnitude of errors between the actual labels and the predicted labels.

---

**Observations from MAE**

For the **sequential neural network**, the calculated MAE was **0.2339**, indicating an average error rate of approximately **23.39%** per sample. After implementing the **ResNet-inspired neural network**, the MAE further decreased to **0.2206**, signifying an improvement in the model's overall prediction accuracy.

This reduction in MAE aligned with the observed enhancements in classification metrics, such as precision, recall, and F1 score, confirming that the ResNet-inspired model was not only better at classification but also at minimizing absolute prediction errors.

---

## Focus on Evaluation Metrics: Recall, Precision, and MAE

Given the significant class imbalance in our dataset, our primary focus was on evaluation metrics that effectively measure the model's performance on the minority class. While overall accuracy provides a general measure of correctness, it can be misleading in imbalanced datasets, as a model can achieve high accuracy by disproportionately predicting the majority class. Therefore, we prioritized **recall**, **precision**, and **mean absolute error (MAE)** for a more comprehensive evaluation.

- **Recall**: This metric was crucial to ensure that the model correctly identified as many positive (minority class) instances as possible. A high recall indicates the model's ability to minimize false negatives, which is critical when the minority class represents an important outcome.

- **Precision**: Precision was equally important to reduce false positives and ensure the model's predictions for the minority class were reliable. This metric was necessary to balance the trade-off between identifying true positives and avoiding over-classification of the minority class.

- **MAE**: Mean Absolute Error provided a holistic view of the model's performance by measuring the average magnitude of prediction errors across all samples. It complemented recall and precision by quantifying the overall error rate and ensuring the model was minimizing deviations in predictions.

By focusing on these metrics, we were able to better address the challenges posed by the imbalanced dataset and validate the model's ability to perform effectively on both classes, with special emphasis on the underrepresented minority class. This approach ensured a robust evaluation that aligned with the objectives of the project.

---

# VALIDATION DATASET EVALUATION

---

To assess the generalization capability of our models, we evaluated both the ResNet-inspired model and the sequential neural network on an unseen validation dataset.

## Validation Dataset Overview

The validation dataset was loaded for evaluation, comprising **X_validation** with its shape determined to be `(n_samples, n_features)` after inspection. The features were standardized using the same `StandardScaler` fitted on the training data to maintain consistency in preprocessing.

---

## ResNet-Inspired Model Prediction

The ResNet-inspired model was applied to the validation dataset to generate predicted probabilities. These probabilities represent the likelihood of each sample belonging to the positive class.

**Steps**:

- The standardized **X_validation** data was passed through the ResNet model to compute predictions.
- The predicted probabilities were stored in a separate DataFrame.

**Output**:

- The predicted probabilities ranged from **0.0 to 1.0**, providing insights into the model's confidence for each prediction.

---

## Sequential Model Prediction

The baseline sequential model was also evaluated on the validation dataset. Similar to the ResNet model, the predicted probabilities for the positive class were calculated.

**Steps**:

- The same standardized **X_validation** was used as input to the baseline sequential model.
- The resulting probabilities were stored in another DataFrame.

**Output**:

- The predicted probabilities for this model were also within the range of **0.0 to 1.0**, offering a basis for comparative evaluation against the ResNet model.

---

## COMPARISON & CONCLUSION

Both models provided outputs in the form of predicted probabilities, allowing for a direct comparison of their performance on unseen data.

Given its ability to effectively handle class imbalance and leverage residual connections for capturing complex patterns, we are selecting the **ResNet-inspired model** for final validation and submission. While both models performed comparably on many metrics, the ResNet-inspired model demonstrated a slight edge in handling the intricacies of the dataset, particularly in scenarios with class imbalance. The baseline sequential model also proved to be a valuable approach during initial exploration and provided reliable predictions, showcasing its potential for similar tasks. However, the ResNet-inspired model's architecture offers additional robustness, making it the preferred choice for this project.