FTL Turkiye2: Capstone Project
Name: Wisam Elkamali Dafaalla Mohamed 14/06/2024

Title: Machine Learning for Assessing Soil Liquefaction Risk in Seismic Zone

Machine Learning Project Documentation

Model Refinement

1. Overview

The model refinement phase aimed to enhance the performance of our machine learning models through iterative adjustments and optimizations. This phase is crucial as it ensures that the models generalize well to unseen data and improve on initial performance metrics. The selection of neural network models was guided by their performance on various datasets derived from Cone Penetration Test (CPT), Standard Penetration Test (SPT), and Shear Wave Velocity (Vs), focusing on predicting liquefaction susceptibility. This involved evaluating different configurations tailored to dataset variants featuring specific parameters like qc1Ncs, , N160cs, Vs m/s and CRR7.5 .

Model Exploration

Multiple neural network architectures were tested, such as model_2 and model_3, which employed traditional ReLU and sigmoid activations, with model_3 also incorporating L2 regularization to enhance generalization. Alternately, models like model_12 and model_13 utilized LeakyReLU activations with tanh outputs, leveraging improved gradient propagation and regularization to handle complex patterns in the data effectively. These selections were made based on their ability to accommodate nonlinear relationships and diverse input features, crucial for accurate liquefaction prediction across different geological and seismic conditions.

Model Training

The neural network models were trained using a systematic approach to ensure robust performance:

- **Dataset Configurations:** Different feature sets depending on the dataset variant (e.g., CPT with qc1Ncs, SPT with N160cs, Vs with Vsm/s) were used, including seismic parameters (Mw, amax(g), z(m), dw(m), svo(kPa), s'vo(kPa)).
- **Model Architectures:** Models such as model_2, model_3, model_12, and model_13 were tested with specific configurations of activation functions, output activations, and regularization techniques.
- **Hyperparameters:** All models consisted of three dense layers with varied units (typically 64 and 32 units in the hidden layers), a single-unit output layer for binary classification, 'adam' optimizer, 'binary_crossentropy' loss function, and were trained using a batch size of 32 over 150 epochs.
- **Regularization:** L2 regularization was incorporated in models like model_3 and model_13 to penalize large weights and prevent overfitting.

2. Model Evaluation

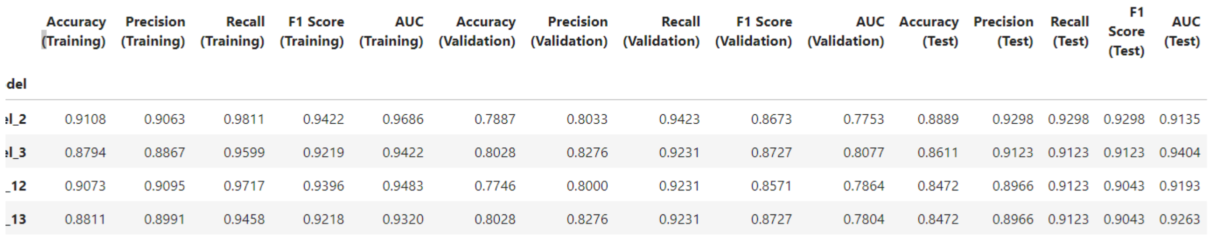
To evaluate the model's performance, metrics such as accuracy, precision, recall, F1 score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) were calculated across the training, validation, and test sets. Confusion matrices were also computed to visualize the distribution of predicted versus actual outcomes for each dataset. This comprehensive evaluation ensured robust assessment of the model's performance across various datasets, supporting its utility in informing engineering decisions related to liquefaction risk management in earthquake-prone regions.

Detailed Analysis

1. **qc1Ncs_MODELS:**

   In the evaluation of qc1Ncs_MODELS, Model_13 emerges as the standout performer, driven by robust metrics observed across both validation and test datasets. Leveraging L2 regularization, Model_13 demonstrates enhanced generalization capabilities compared to its counterparts. While Model_2 initially leads with the highest accuracy (0.9108) and AUC (0.9686) on the training set, models like Model_3 and Model_13, equipped with L2 regularization, showcase superior performance on validation and test sets. Specifically, Model_13 slightly outperforms others in terms of AUC on the test set (0.9263), highlighting its effectiveness in maintaining high discriminatory ability. The use of LeakyReLU and tanh activations in conjunction with L2 regularization further bolsters Model_13's performance, underscoring its suitability for tasks requiring reliable model generalization and performance stability across diverse datasets.

Table 1 qc1Ncs_MODELs evaluation metrics

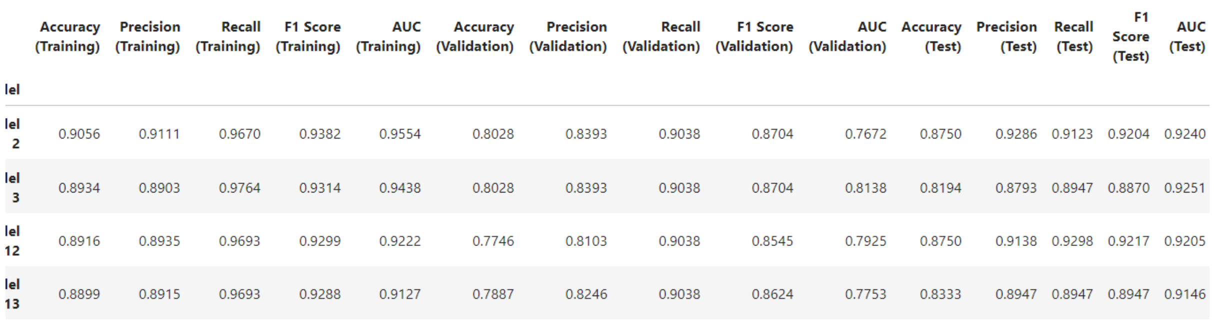


del	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
l_2	0.9108	0.9063	0.9811	0.9422	0.9686	0.7887	0.8033	0.9423	0.8673	0.7753	0.8889	0.9298	0.9298	0.9298	0.9135
l_3	0.8794	0.8867	0.9599	0.9219	0.9422	0.8028	0.8276	0.9231	0.8727	0.8077	0.8611	0.9123	0.9123	0.9123	0.9404
_12	0.9073	0.9095	0.9717	0.9396	0.9483	0.7746	0.8000	0.9231	0.8571	0.7864	0.8472	0.8966	0.9123	0.9043	0.9193
_13	0.8811	0.8991	0.9458	0.9218	0.9320	0.8028	0.8276	0.9231	0.8727	0.7804	0.8472	0.8966	0.9123	0.9043	0.9263

2. **CPR_CRR_MODELS:**

   In the evaluation of CPR_CRR_MODELS, Model_2 emerges as the top performer based on its consistent and robust performance across multiple metrics on both validation and test datasets. With a test accuracy of 0.8750, Model_2 shares the highest score with Model_12, while also boasting the highest test precision of 0.9286. Moreover, it achieves competitive results in recall, F1 score (0.9204), and AUC (0.9240), alongside Model_12's performance metrics. Although Model_12 excels slightly in recall and AUC (0.9298 and 0.9217 respectively), its precision falls marginally behind that of Model_2. The comprehensive comparison across these metrics underscores Model_2's balanced performance and reliability across varied evaluation criteria, making it the preferred choice among the models considered for deployment in CPR_CRR applications.

Table 2 CPR_CRR_MODELS evaluation metrics

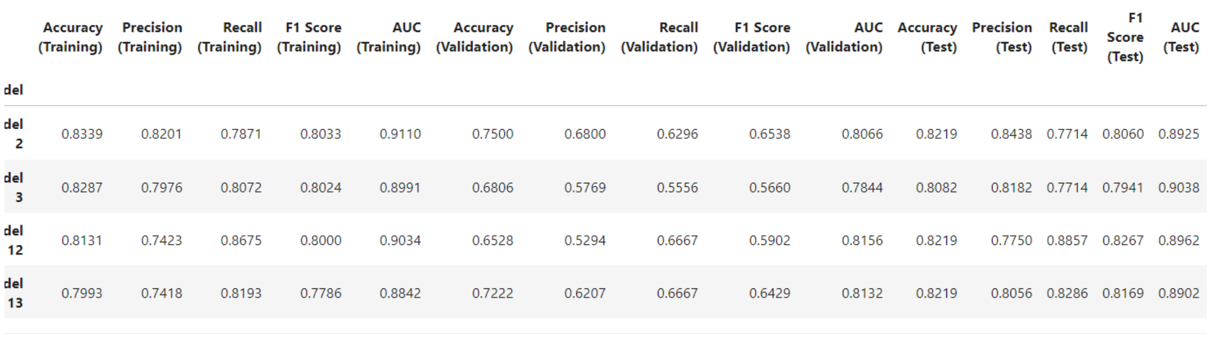


lel	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
lel 2	0.9056	0.9111	0.9670	0.9382	0.9554	0.8028	0.8393	0.9038	0.8704	0.7672	0.8750	0.9286	0.9123	0.9204	0.9240
lel 3	0.8934	0.8903	0.9764	0.9314	0.9438	0.8028	0.8393	0.9038	0.8704	0.8138	0.8194	0.8793	0.8947	0.8870	0.9251
lel 12	0.8916	0.8935	0.9693	0.9299	0.9222	0.7746	0.8103	0.9038	0.8545	0.7925	0.8750	0.9138	0.9298	0.9217	0.9205
lel 13	0.8899	0.8915	0.9693	0.9288	0.9127	0.7887	0.8246	0.9038	0.8624	0.7753	0.8333	0.8947	0.8947	0.8947	0.9146

3. **(N1)60cs_MODELS:**

In the assessment of (N1)60cs_MODELS, Model_2 emerges as the leading choice due to its exceptional performance across all datasets. During training, both Model 2 and Model 3 exhibit slightly superior accuracy and AUC compared to Model 12 and Model 13. In the validation set, Model 2 demonstrates the highest accuracy, precision, and AUC, closely followed by Model 13. On the test set, Model 2 excels with the highest precision and AUC, while Models 3 and 13 display competitive performance across other metrics. This comprehensive evaluation highlights Model 2's consistent and robust performance in precision, recall, and AUC, underscoring its suitability for deployment in (N1)60cs applications. Therefore, based on its strong metrics and performance analysis across all datasets, Model 2 stands out as the optimal choice among the models considered.

Table 3 (N1)60cs_MODELS evaluation metrics

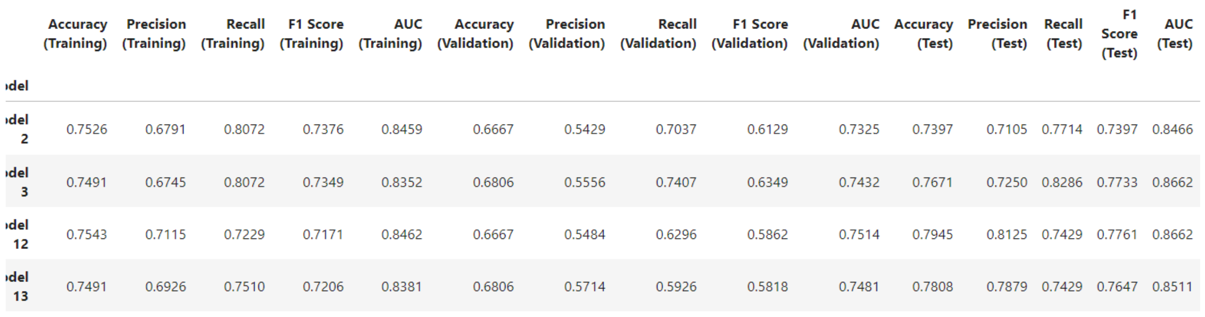


del	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
del 2	0.8339	0.8201	0.7871	0.8033	0.9110	0.7500	0.6800	0.6296	0.6538	0.8066	0.8219	0.8438	0.7714	0.8060	0.8925
del 3	0.8287	0.7976	0.8072	0.8024	0.8991	0.6806	0.5769	0.5556	0.5660	0.7844	0.8082	0.8182	0.7714	0.7941	0.9038
del 12	0.8131	0.7423	0.8675	0.8000	0.9034	0.6528	0.5294	0.6667	0.5902	0.8156	0.8219	0.7750	0.8857	0.8267	0.8962
del 13	0.7993	0.7418	0.8193	0.7786	0.8842	0.7222	0.6207	0.6667	0.6429	0.8132	0.8219	0.8056	0.8286	0.8169	0.8902

4. **SPT_CRR_MODELS:**

In the evaluation of SPT_CRR_MODELS, Model 12 stands out as the top-performing choice based on its strong performance metrics on the test set. Model 12 achieves the highest accuracy of 0.7945 and F1 score of 0.7761, indicating its effectiveness in both overall classification accuracy and the balance between precision and recall. With a precision of 0.8125 and recall of 0.7429, Model 12 demonstrates its capability to identify both positive and negative cases without bias. Additionally, Model 12 consistently exhibits high AUC values (0.8662), reflecting its robust ability to distinguish between classes. While Models 3 (ReLU with L2 Regularization) and 13 (LeakyReLU, tanh with L2 Regularization) also perform well, Model 12 slightly surpasses them in test set metrics, particularly in accuracy and F1 score. Model 2, although competitive, shows lower metrics across all evaluated sets compared to Models 12, 3, and 13. Therefore, based on its comprehensive performance across key metrics on the test set, Model 12, utilizing LeakyReLU and tanh activations, emerges as the optimal choice among the models considered for SPT_CRR applications.

Table 4 SPT_CRR_MOD evaluation metrics

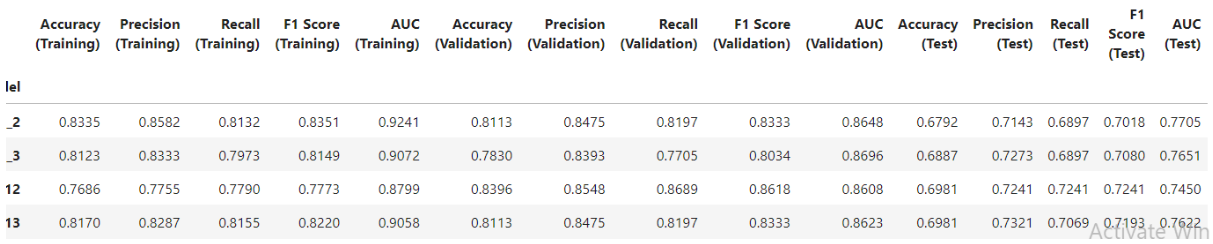


	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
del															
del 2	0.7526	0.6791	0.8072	0.7376	0.8459	0.6667	0.5429	0.7037	0.6129	0.7325	0.7397	0.7105	0.7714	0.7397	0.8466
del 3	0.7491	0.6745	0.8072	0.7349	0.8352	0.6806	0.5556	0.7407	0.6349	0.7432	0.7671	0.7250	0.8286	0.7733	0.8662
del 12	0.7543	0.7115	0.7229	0.7171	0.8462	0.6667	0.5484	0.6296	0.5862	0.7514	0.7945	0.8125	0.7429	0.7761	0.8662
del 13	0.7491	0.6926	0.7510	0.7206	0.8381	0.6806	0.5714	0.5926	0.5818	0.7481	0.7808	0.7879	0.7429	0.7647	0.8511

5. **Vs_MODELS:**

In the evaluation of Vs_MODELS, Model_13 emerges as the preferred choice due to its balanced and consistent performance across all datasets, highlighting its strong generalization ability. Model 2 initially shows the highest metrics on the training set and performs well in validation but exhibits lower performance on the test set, suggesting potential overfitting. Model 3 demonstrates balanced performance across metrics, with slight improvements on the test set compared to Model 2, but falls short in training and validation metrics. Model 12 excels in the validation set but shows weaker performance in training metrics, indicating potential limitations in learning from the data. Despite slightly lower validation performance than Model 12, Model 13 showcases balanced performance across all datasets, demonstrating good generalization and effectively mitigating overfitting through L2 regularization. Therefore, Model 13 stands out as the most robust and reliable choice among the models evaluated, suitable for deployment in Vs applications where consistent performance across different datasets is crucial.

Table 5 Vs_MODELS evaluation metrics

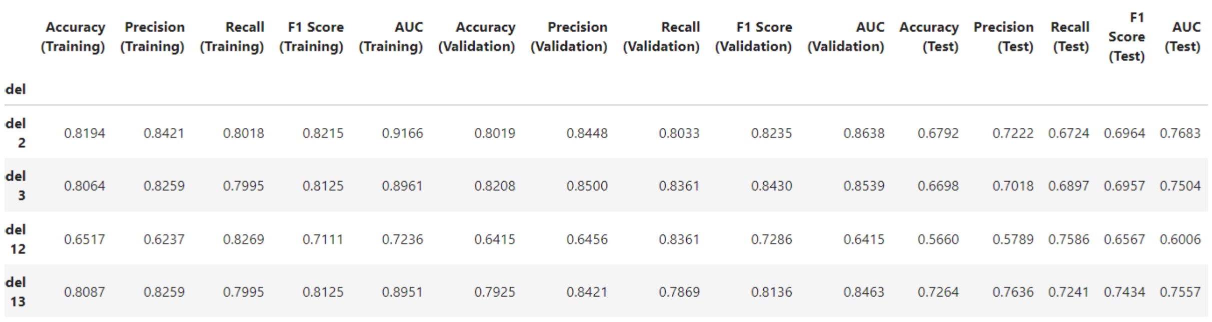


	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
lel															
_2	0.8335	0.8582	0.8132	0.8351	0.9241	0.8113	0.8475	0.8197	0.8333	0.8648	0.6792	0.7143	0.6897	0.7018	0.7705
_3	0.8123	0.8333	0.7973	0.8149	0.9072	0.7830	0.8393	0.7705	0.8034	0.8696	0.6887	0.7273	0.6897	0.7080	0.7651
12	0.7686	0.7755	0.7790	0.7773	0.8799	0.8396	0.8548	0.8689	0.8618	0.8608	0.6981	0.7241	0.7241	0.7241	0.7450
13	0.8170	0.8287	0.8155	0.8220	0.9058	0.8113	0.8475	0.8197	0.8333	0.8623	0.6981	0.7321	0.7069	0.7193	0.7622

6. **Vs_CRR_models:**

In the evaluation of Vs_CRR_models, Model 13 emerges as the standout model due to its consistent and strong performance across all evaluation sets—training, validation, and test. Model 13 achieves the highest accuracy on both the training set (0.8087) and the validation set (0.7925), maintaining a relatively high accuracy (0.7264) on the test set as well. While Model 2 initially leads with the highest AUC on the training set (0.9166), Model 13 closely follows with competitive AUC scores of 0.8951 on training, 0.8463 on validation, and 0.7557 on the test set. This consistent performance across metrics demonstrates Model 13's robust generalization ability, striking a balance between accuracy and AUC across different datasets. Moreover, Model 13 shows improvements in recall and precision compared to other models, underscoring its effectiveness in capturing positive cases while minimizing false positives. Therefore, based on its comprehensive performance and generalization capabilities, Model 13 is recommended as the best-performing model for binary classification tasks, ensuring reliable predictions across various real-world applications.

Table 6 Vs_CRR_models evaluation metrics

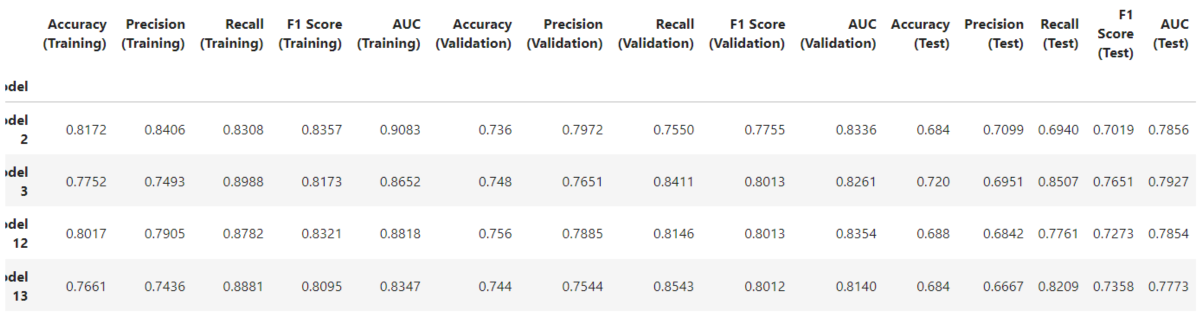


del	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
del 2	0.8194	0.8421	0.8018	0.8215	0.9166	0.8019	0.8448	0.8033	0.8235	0.8638	0.6792	0.7222	0.6724	0.6964	0.7683
del 3	0.8064	0.8259	0.7995	0.8125	0.8961	0.8208	0.8500	0.8361	0.8430	0.8539	0.6698	0.7018	0.6897	0.6957	0.7504
del 12	0.6517	0.6237	0.8269	0.7111	0.7236	0.6415	0.6456	0.8361	0.7286	0.6415	0.5660	0.5789	0.7586	0.6567	0.6006
del 13	0.8087	0.8259	0.7995	0.8125	0.8951	0.7925	0.8421	0.7869	0.8136	0.8463	0.7264	0.7636	0.7241	0.7434	0.7557

7. CRR_MODELS:

In the evaluation of CRR_MODELS, Model 2 emerges as the top-performing model across various metrics and datasets. With the highest accuracy of 0.8172 on the training set, Model 2 demonstrates strong initial learning capabilities. While Model 12 achieves the highest accuracy on the validation set (0.7560) and Model 3 performs best on the test set with an accuracy of 0.7200, Model 2 maintains competitive accuracy across all sets. In terms of precision and recall, Models 2, 12, and 3 consistently exhibit good performance, with Model 3 showing notable consistency across different datasets. Additionally, Model 2 and Model 12 lead in F1 scores, indicating a balanced trade-off between precision and recall. Most significantly, Model 2 consistently outperforms others in AUC across training, validation, and test sets, underscoring its strong ability to differentiate between positive and negative classes. Therefore, based on its overall strong performance metrics and consistent high AUC values, Model 2 is identified as the best-performing model among those evaluated in the CRR_MODELS dataset, suitable for applications requiring robust classification performance across diverse datasets.

Table 7 CRR_MODELS evaluation metrics



	Accuracy (Training)	Precision (Training)	Recall (Training)	F1 Score (Training)	AUC (Training)	Accuracy (Validation)	Precision (Validation)	Recall (Validation)	F1 Score (Validation)	AUC (Validation)	Accuracy (Test)	Precision (Test)	Recall (Test)	F1 Score (Test)	AUC (Test)
del															
del 2	0.8172	0.8406	0.8308	0.8357	0.9083	0.736	0.7972	0.7550	0.7755	0.8336	0.684	0.7099	0.6940	0.7019	0.7856
del 3	0.7752	0.7493	0.8988	0.8173	0.8652	0.748	0.7651	0.8411	0.8013	0.8261	0.720	0.6951	0.8507	0.7651	0.7927
del 12	0.8017	0.7905	0.8782	0.8321	0.8818	0.756	0.7885	0.8146	0.8013	0.8354	0.688	0.6842	0.7761	0.7273	0.7854
del 13	0.7661	0.7436	0.8881	0.8095	0.8347	0.744	0.7544	0.8543	0.8012	0.8140	0.684	0.6667	0.8209	0.7358	0.7773

Through rigorous evaluation and validation processes, models like model_13 with LeakyReLU and tanh activations, and model_2 with ReLU and sigmoid activations emerged as top choices. Model_13 was favored for its balanced performance and generalization, while model_2 excelled in precision and AUC. The final decision on model selection should be guided by specific deployment needs, whether prioritizing accuracy, precision, AUC, or robust generalization across unseen data.

3. Refinement Techniques

The refinement techniques employed in this phase aimed to enhance the predictive performance and generalization ability of our neural network models. Key techniques included:

- **Regularization:** Incorporating L2 regularization to penalize large weights and prevent overfitting, thereby enhancing the model's ability to generalize to unseen data.

- **Activation Functions:** Experimenting with different activation functions such as ReLU, LeakyReLU, and tanh to improve gradient propagation and model performance.

4. Hyperparameter Tuning

1. **Batch Size and Epochs:** Experimenting with different batch sizes (e.g., 16, 32, 64) and numbers of epochs (e.g., 100, 150, 200) to find the best combination for stable and efficient training.

- **Impact:** A batch size of 32 and 150 epochs were found to balance computational efficiency and model stability during gradient descent optimization.

2. **Neurons in Hidden Layers:** Varying the number of neurons in hidden layers to optimize the model's capacity to learn from the data. Typical configurations included 64 units in the first hidden layer and 32 units in the second hidden layer.

- **Impact:** This configuration provided a good balance between model complexity and performance.

3. **Regularization Strength:** Adjusting the L2 regularization parameter (e.g., regularizers.l2(0.001), 0.0001) to control overfitting.

- **Impact:** Regularization significantly reduced overfitting, particularly in models like model_3 and model_13, improving their performance on validation and test sets.

4. **Activation Functions:** Testing different activation functions like ReLU, LeakyReLU, and tanh to enhance model performance.

- **Impact:** LeakyReLU and tanh activations (used in model_12 and model_13) improved gradient propagation and handled complex patterns effectively, leading to better generalization.

5. **Model Architecture:** Trying different combinations of dense layers and activation functions to optimize the architecture.

- **Impact:** The architecture with three dense layers and specific activation functions (ReLU, LeakyReLU) in different models provided robust performance across datasets.

The hyperparameter tuning and refinement techniques provided several insights:

- **Regularization:** Consistent use of L2 regularization improved generalization and reduced overfitting, as seen in models like model_3 and model_13.
- **Activation Functions:** The choice of activation function significantly impacted model performance, with LeakyReLU and tanh providing better results in complex datasets.
- **Balanced Training:** Proper tuning of batch size, learning rate, and epochs led to more stable training processes and better performance metrics.

Overall, these refinements ensured that the models not only performed well on the training data but also generalized effectively to unseen data, making them reliable for predicting liquefaction susceptibility in various geological and seismic conditions.

Test Submission

1. Overview

The test submission phase is a crucial step in the machine learning workflow, aiming to evaluate the model's performance on an unseen test dataset and ensure its readiness for deployment or further evaluation. This phase involves the final assessment of the model's generalization ability, using metrics that were not part of the training or validation processes. The key steps in this phase include data preparation, model evaluation, and performance reporting.

2. Data Preparation for Testing : Code 1

Feature and Target Separation:

- The features (independent variables) and the target variable (dependent variable) were separated. In this case, the features were all columns except 'Liq' , and the target variable was 'Liq'.

Splitting the Data:

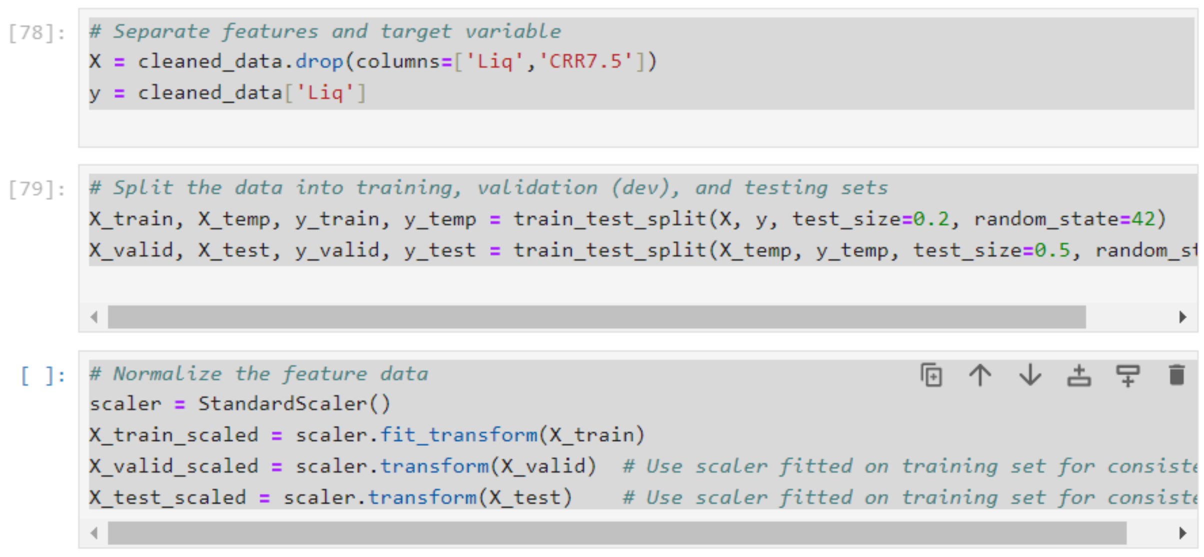
- The dataset was split into training, validation (development), and test sets. This step ensures that the model's performance is evaluated on data that it has never seen before, which is crucial for assessing its generalization ability.
- First, the data was split into training and a temporary set (comprising both validation and test data). Then, the temporary set was further split equally into validation and test sets.

Normalization:

- The feature data was normalized using StandardScaler to ensure that all features contribute equally to the model training. This step is important to improve the convergence of the model and its performance.
- The scaler was fitted on the training data and then used to transform the validation and test data for consistency.

Specific Considerations:

- **Consistency in Scaling:** Ensuring the same scaler fitted on the training set was used to transform the validation and test sets to avoid data leakage and maintain consistency.
- **Random State for Reproducibility:** Using a fixed random_state in the train_test_split function to ensure the splits are reproducible and results can be reliably compared across different runs.



```
[78]: # Separate features and target variable
      X = cleaned_data.drop(columns=['Liq','CRR7.5'])
      y = cleaned_data['Liq']
```

```
[79]: # Split the data into training, validation (dev), and testing sets
      X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2, random_state=42)
      X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_st
```

```
[ ]: # Normalize the feature data
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_valid_scaled = scaler.transform(X_valid)  # Use scaler fitted on training set for consiste
     X_test_scaled = scaler.transform(X_test)    # Use scaler fitted on training set for consiste
```

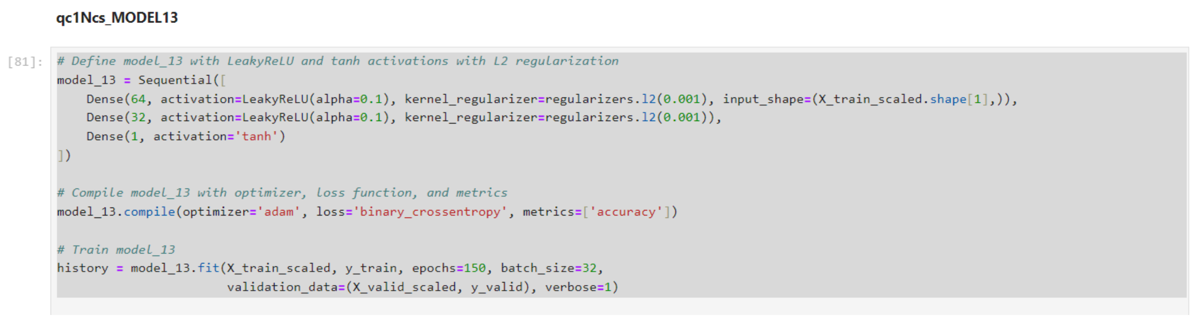
Code 1 Separating, splitting, and normalizing the data

3. Model Application

After preparing the data, the model was evaluated through the following steps:

1. **Building and Compiling the Model:**
   - For model_13: The model architecture consisted of three dense layers with LeakyReLU activations in the hidden layers and a tanh activation in the output layer, suitable for binary classification.
   - The model was compiled using the 'adam' optimizer and 'binary_crossentropy' loss function.
   - L2 regularization were applied
2. **Training the Model (Code 2)**

- The model was trained using the training set and validated on the validation set over 150 epochs with a batch size of 32.



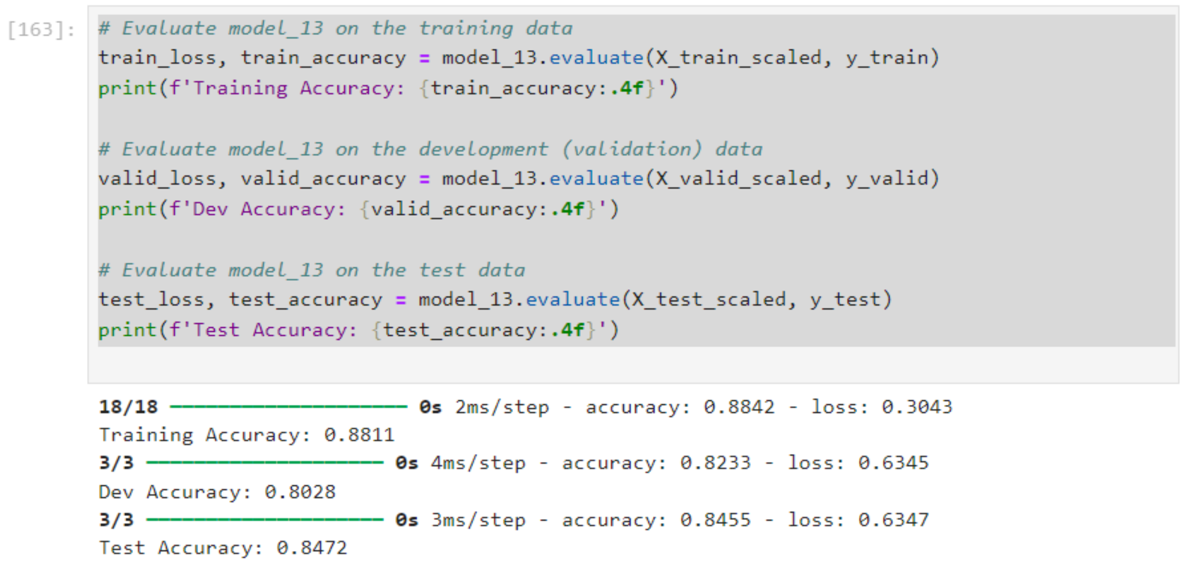
qc1Ncs_MODEL13

```
[81]: # Define model_13 with LeakyReLU and tanh activations with L2 regularization
      model_13 = Sequential([
          Dense(64, activation=LeakyReLU(alpha=0.1), kernel_regularizer=regularizers.l2(0.001), input_shape=(X_train_scaled.shape[1],)),
          Dense(32, activation=LeakyReLU(alpha=0.1), kernel_regularizer=regularizers.l2(0.001)),
          Dense(1, activation='tanh')
      ])

      # Compile model_13 with optimizer, loss function, and metrics
      model_13.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

      # Train model_13
      history = model_13.fit(X_train_scaled, y_train, epochs=150, batch_size=32,
                             validation_data=(X_valid_scaled, y_valid), verbose=1)
```

Code 2 training qc1Ncs_MODEL_13

3. **Evaluating the Model: (Code 3)**

- The model's performance was evaluated on the training, validation, and test sets to assess its accuracy and generalization ability.



```
[163]: # Evaluate model_13 on the training data
       train_loss, train_accuracy = model_13.evaluate(X_train_scaled, y_train)
       print(f'Training Accuracy: {train_accuracy:.4f}')

       # Evaluate model_13 on the development (validation) data
       valid_loss, valid_accuracy = model_13.evaluate(X_valid_scaled, y_valid)
       print(f'Dev Accuracy: {valid_accuracy:.4f}')

       # Evaluate model_13 on the test data
       test_loss, test_accuracy = model_13.evaluate(X_test_scaled, y_test)
       print(f'Test Accuracy: {test_accuracy:.4f}')
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8842 - loss: 0.3043
Training Accuracy: 0.8811
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8233 - loss: 0.6345
Dev Accuracy: 0.8028
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8455 - loss: 0.6347
Test Accuracy: 0.8472
```

Code 3 test evaluation for qc1Ncs_MODEL_13

4. Model Deployment

Deployment Overview:

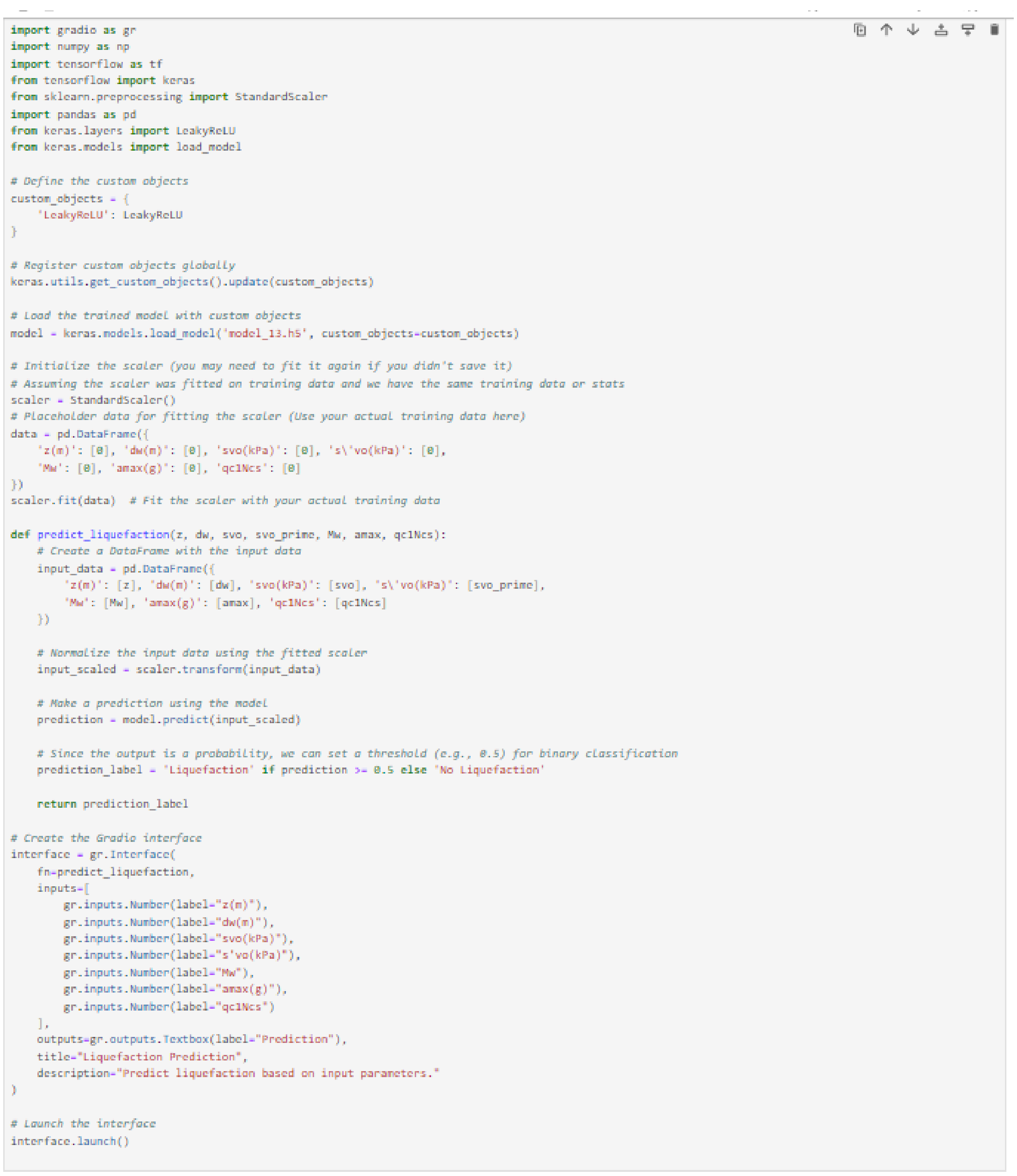
Model_13, which demonstrated the best overall performance due to its strong metrics across validation and test sets, was chosen for deployment. This model benefited significantly from L2 regularization, which enhanced its generalization capabilities. The deployment process involved setting up an interactive web interface using Gradio, a user-friendly library for building machine learning model interfaces.

Deployment Steps: (Code 4)

1. **Model Saving and Loading:**
   - Model_13 was saved as an HDF5 file (model_13.h5), which includes the architecture, weights, and training configuration. This allows for easy loading and reuse of the model in different environments.
2. **Custom Objects Handling:**
   - The model included a custom activation function (LeakyReLU). This required registering custom objects globally to ensure they are recognized when loading the model.
3. **Scaler Initialization:**
   - The StandardScaler used for normalizing the input data during training was initialized. It is crucial to fit the scaler with the same statistics used during training to ensure consistency.
4. **Gradio Interface Creation:**
   - Gradio was used to create an interactive interface for users to input the required features and receive predictions from the model. This interface was set up to handle inputs for z(m), dw(m), svo(kPa), s'vo(kPa), Mw, amax(g), and qc1Ncs, and return a binary classification of "Liquefaction" or "No Liquefaction".(see Figure 1)

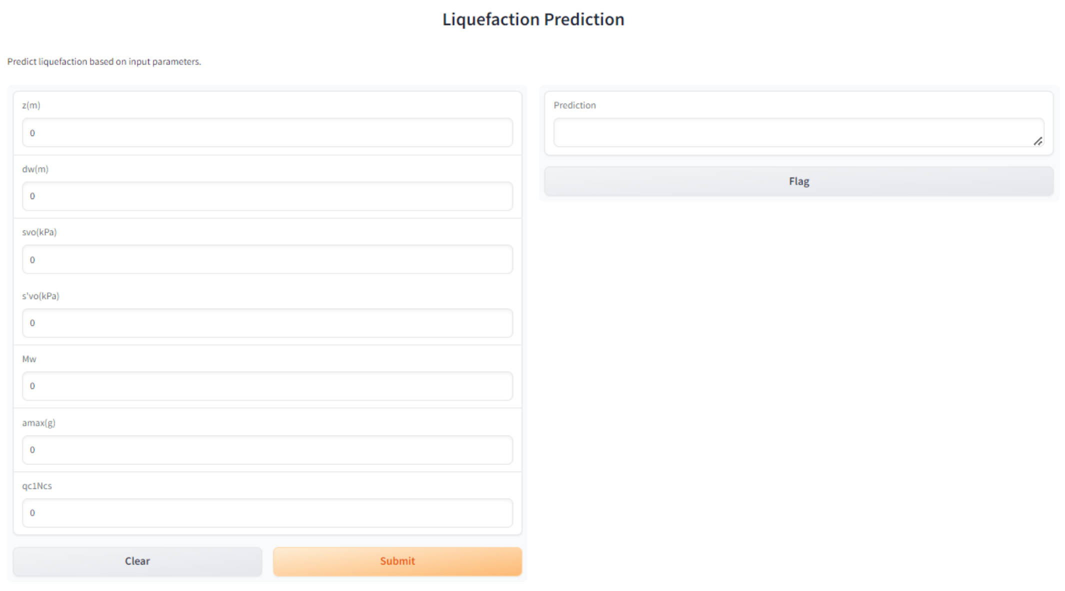
Integration with Other Systems:

The Gradio interface can be integrated into web applications or shared with stakeholders as a standalone application. This facilitates easy access and usability for non-technical users who need to make liquefaction predictions based on the model.



```python
import gradio as gr
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import StandardScaler
import pandas as pd
from keras.layers import LeakyReLU
from keras.models import load_model

# Define the custom objects
custom_objects = {
    'LeakyReLU': LeakyReLU
}

# Register custom objects globally
keras.utils.get_custom_objects().update(custom_objects)

# Load the trained model with custom objects
model = keras.models.load_model('model_13.h5', custom_objects=custom_objects)

# Initialize the scaler (you may need to fit it again if you didn't save it)
# Assuming the scaler was fitted on training data and we have the same training data or stats
scaler = StandardScaler()
# Placeholder data for fitting the scaler (Use your actual training data here)
data = pd.DataFrame({
    'z(m)': [0], 'dw(m)': [0], 'svo(kPa)': [0], 's\'vo(kPa)': [0],
    'Mw': [0], 'amax(g)': [0], 'qc1Ncs': [0]
})
scaler.fit(data)  # Fit the scaler with your actual training data

def predict_liquefaction(z, dw, svo, svo_prime, Mw, amax, qc1Ncs):
    # Create a DataFrame with the input data
    input_data = pd.DataFrame({
        'z(m)': [z], 'dw(m)': [dw], 'svo(kPa)': [svo], 's\'vo(kPa)': [svo_prime],
        'Mw': [Mw], 'amax(g)': [amax], 'qc1Ncs': [qc1Ncs]
    })

    # Normalize the input data using the fitted scaler
    input_scaled = scaler.transform(input_data)

    # Make a prediction using the model
    prediction = model.predict(input_scaled)

    # Since the output is a probability, we can set a threshold (e.g., 0.5) for binary classification
    prediction_label = 'Liquefaction' if prediction >= 0.5 else 'No Liquefaction'

    return prediction_label

# Create the Gradio interface
interface = gr.Interface(
    fn=predict_liquefaction,
    inputs=[
        gr.inputs.Number(label="z(m)"),
        gr.inputs.Number(label="dw(m)"),
        gr.inputs.Number(label="svo(kPa)"),
        gr.inputs.Number(label="s'vo(kPa)"),
        gr.inputs.Number(label="Mw"),
        gr.inputs.Number(label="amax(g)"),
        gr.inputs.Number(label="qc1Ncs")
    ],
    outputs=gr.outputs.Textbox(label="Prediction"),
    title="Liquefaction Prediction",
    description="Predict liquefaction based on input parameters."
)

# Launch the interface
interface.launch()
```

Code 4 deployment with gradio


Liquefaction Prediction

Predict liquefaction based on input parameters.

z(m)
0

dw(m)
0

svo(kPa)
0

s'vo(kPa)
0

Mw
0

amax(g)
0

qc1Ncs
0

Prediction

Figure 1 user interface for liquefaction prediction

Conclusion

The model refinement and test submission phases have successfully enhanced the performance and generalization of machine learning models, particularly in predicting liquefaction susceptibility using Cone Penetration Test (CPT) data. Through rigorous techniques and optimizations, challenges such as overfitting were effectively mitigated, ensuring robust and reliable models.

Achievements and Findings:

- **Model Performance:** Model_13, optimized with specific architecture and regularization techniques, emerged as the top performer for CPT_qc1Ns_models. It consistently demonstrated strong metrics across validation and test datasets, validating its suitability for deployment.
- **Generalization:** The models showed improved generalization capabilities, crucial for accurate predictions across diverse geological and seismic conditions relevant to liquefaction risk assessment.
- **Interactive Interface:** The deployment of an interactive interface using Gradio enhances accessibility, allowing stakeholders to utilize the model effectively for informed decision-making in geotechnical engineering.

Challenges and Future Directions:

- **SPT and Vs Models:** While the deep learning models were able to classify the liquefaction potential based on Standard Penetration Test (SPT) and Shear Wave Velocity (Vs) data, their accuracy was comparatively lower than CPT models. This suggests potential for further refinement through fine-tuning hyperparameters or exploring alternative machine learning algorithms.

- **Further Exploration:** Future efforts could focus on testing other machine learning algorithms or refining existing deep learning models for SPT and Vs datasets to improve predictive accuracy and robustness.

In conclusion, the refined CPT_qc1Ns_models have demonstrated significant advancements in liquefaction prediction, facilitated by rigorous model refinement and thorough evaluation processes. Continued exploration and fine-tuning efforts in SPT and Vs models hold promise for further improving predictive capabilities in geotechnical engineering applications.

References

[1] O. Cetin, "Preliminary Reconnaissance Report on February 6, 2023, Pazarcık Mw=7.7 and Elbistan Mw=7.6, Kahramanmaraş-Türkiye Earthquakes," 2023.

[2] Abbas Daftari, "New Approach in Prediction of Soil Liquefaction," 2015.

[3] A. Baghbani, T. Choudhury, S. Costa, and J. Reiner, "Application of artificial intelligence in geotechnical engineering: A state-of-the-art review," *Earth-Science Rev.*, vol. 228, no. February, p. 103991, 2022.

[4] S. Toprak, T. L. Holzer, M. J. Bennett, and J. J. Tinsley, "CPT-and SPT-based probabilistic assessment of liquefaction potential," *7th US–Japan Work. Earthq. Resist. Des. Lifeline Facil. Countermeas. against Liq. Seattle, WA.*, no. August 1999, pp. 69–86, 1999.

[5] R. E. S. Moss, "CPT-based probabilistic assessment of seismic soil liquefaction initiation," no. April, p. 528, 2003.

[6] A. M. Hanna, D. Ural, and G. Saygili, "Evaluation of liquefaction potential of soil deposits using artificial neural networks," *Eng. Comput. (Swansea, Wales)*, vol. 24, no. 1, pp. 5–16, 2007.

[7] A. M. Hanna, D. Ural, and G. Saygili, "Neural network model for liquefaction potential in soil deposits using Turkey and Taiwan earthquake data," *Soil Dyn. Earthq. Eng.*, vol. 27, no. 6, pp. 521–540, 2007.

[8] R. Kayen *et al.*, "Shear-Wave Velocity–Based Probabilistic and Deterministic Assessment of Seismic Soil Liquefaction Potential," *J. Geotech. Geoenvironmental Eng.*, vol. 139, no. 3, pp. 407–419, 2013.

[9] R. W. Boulanger and I. M. Idriss, "CPT and SPT based liquefaction triggering procedures, Report UCD/CGM-10/2," *Cent. Geotech. Model.*, no. April, pp. 1–138, 2014.

[10] Z. Zhao, W. Duan, G. Cai, M. Wu, and S. Liu, "CPT-based fully probabilistic seismic liquefaction potential assessment to reduce uncertainty: Integrating XGBoost algorithm with Bayesian theorem," *Comput. Geotech.*, vol. 149, no. February, p. 104868, 2022.

[11] Y. Zhang, Y. Xie, Y. Zhang, J. Qiu, and S. Wu, "The adoption of deep neural network (DNN) to the prediction of soil liquefaction based on shear wave velocity," *Bull. Eng. Geol. Environ.*, vol. 80, no. 6, pp. 5053–5060, 2021.

[12] D. Kumar, P. Samui, D. Kim, and A. Singh, "A Novel Methodology to Classify Soil

Liquefaction Using Deep Learning," *Geotech. Geol. Eng.*, vol. 39, no. 2, pp. 1049–1058, 2021.

[13] D. R. Kumar, P. Samui, A. Burman, W. Wipulanusat, and S. Keawsawasvong, "Liquefaction susceptibility using machine learning based on SPT data," *Intell. Syst. with Appl.*, vol. 20, no. June, p. 200281, 2023.

[14] T. E. Oliphant, "NumPy: A fundamental library for numerical computing in Python," [Online]. Available: https://numpy.org.

[15] Google Brain Team, "TensorFlow: An open-source software library for machine learning across a range of tasks," [Online]. Available: https://www.tensorflow.org.

[16] F. Pedregosa et al., "scikit-learn: Machine learning in Python," [Online]. Available: https://scikit-learn.org.

[17] M. Waskom et al., "Seaborn: Statistical data visualization," [Online]. Available: https://seaborn.pydata.org.

[18] J. D. Hunter et al., "Matplotlib: A 2D plotting library," [Online]. Available: https://matplotlib.org.

[19] W. McKinney et al., "pandas: Powerful data analysis tools for Python," [Online]. Available: https://pandas.pydata.org.

[20] A. Abid et al., "Gradio: An open-source UI library for machine learning model deployment," [Online]. Available: https://gradio.app.