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

# Data Preparation/Feature Engineering

## 1. Overview

The data preparation and feature engineering phase is critical in any machine learning project, especially when assessing liquefaction potential in soil. This phase involves transforming raw data from diverse sources such as geotechnical reports, seismic surveys, and academic papers into a structured format suitable for analysis and modeling. The significance of this phase lies in its ability to ensure data quality, relevance, and consistency, which are essential for building robust predictive models.

## 2. Data Collection

Data sources include geotechnical reports, seismic surveys, and academic research papers. Data formats are primarily tabular datasets and research documents with liquefied and non-liquefied data cases, with data sizes varying based on the specific seismic events and regions analyzed. The selection of this data is due to its relevance in assessing liquefaction potential, providing a comprehensive understanding of the contributing factors: Those data sets include:

- Toprak (1999)[1] : CPT data set, SPT data set
- Moss(2006)[2] : CPT data set
- Hanna (2007)[3]: CPT and Vs data set
- Hanna (2007)[4]:SPT and Vs data set
- Kayen (2013) [5]: Vs data set
- Boulanger(2014)[6] : CPT data set, SPT data set
- Zhao (2022) [7]: CPT data set

## 3. Data Cleaning

Data cleaning involved several systematic steps to ensure the datasets were ready for analysis. Initially, taking the CPT data sets as an example; the datasets from five different Excel files were imported. For each dataset, the data types of the columns were examined, and necessary conversions were made, such as transforming the 'Liq' column in `data_1` from string values ('yes', 'no') to binary numeric values (1, 0). Missing values were handled by identifying and dropping rows where crucial columns had invalid or null entries. Outliers were managed by ensuring all entries in the 'Liq' column were valid binary values, removing any rows that did not comply. Common columns across all datasets were identified to facilitate merging, and unique columns specific to individual datasets were removed. The datasets were then merged into a single data frame. After merging, rows containing any NaN values were dropped to ensure completeness. Finally, a new column 'CRR7.5' was calculated based on a provided formula, ensuring all values within the expected range were handled correctly, and invalid entries were flagged. This comprehensive approach ensured a clean and consistent dataset for subsequent analysis. The data preprocessing is shown in Code 1, Code 2 and Code 3.

[7]:
```
path_1 = 'Boulanger_2014_CPT.xlsx'
path_2 = 'Hanna_2006_CPT_Vs.xlsx'
path_3 = 'Moss_2006_CPT.xlsx'
path_4 = 'Toprak_1999_CPT.xlsx'
path_5 = 'Zhao_2022_CPT.xlsx'
```

[8]:
```
# Importing the datasets
data_1 = pd.read_excel(path_1)
data_2 = pd.read_excel(path_2)
data_3 = pd.read_excel(path_3)
data_4 = pd.read_excel(path_4)
data_5 = pd.read_excel(path_5)
```

*Code 1 Importing the CPT data sets*

[12]:
```
## Check data type of 'Liq' column
print("Data type of 'Liq' column:", data_1['Liq'].dtype)

# Convert to string if necessary
data_1['Liq'] = data_1['Liq'].astype(str)

# Convert all values to lowercase
data_1['Liq'] = data_1['Liq'].str.lower()

# Replace 'yes' with 1 and 'no' with 0
data_1['Liq'] = data_1['Liq'].replace({'yes': 1, 'no': 0})

# Explicitly call result.infer_objects(copy=False) to avoid future warning
data_1['Liq'] = data_1['Liq'].infer_objects(copy=False)

# Print the modified DataFrame
data_1
```

*Code 2 Cleaning 'Liq' column in data_1 and encoding (yes, no) to (0,1)*

[55]:
```
# Chain merge operations
merged_data_1_2_3_4_5 = (
    data_1_copy.merge(data_2_copy, how='outer')
               .merge(data_3_copy, how='outer')
               .merge(data_4_copy, how='outer')
               .merge(data_5_copy, how='outer')
)

# Display the result
merged_data_1_2_3_4_5
```

*Code 3 Merging datasets on common columns*

## 4. Exploratory Data Analysis (EDA)

The Exploratory Data Analysis (EDA) phase involved examining the cleaned dataset through various visualizations and statistical measures to uncover key insights. Boxplots (Figure 1) were created for columns such as 'Mw', 'amax(g)', 'Liq', 'z(m)', 'dw(m)', 'svo(kPa)', 's'vo(kPa)', 'qc1Ncs', and 'CRR7.5' to identify the presence of outliers and understand the distribution of these variables.

The box plot visualization provides valuable insights into the distribution and spread of several key variables in the dataset. Each box plot shows the median, quartiles, and potential outliers for columns such as 'Mw', 'amax(g)', 'Liq', 'z(m)', 'dw(m)', 'svo(kPa)', 's'vo(kPa)', 'qc1Ncs', and 'CRR7.5'. Here are some observations:

1. **Mw (Moment Magnitude):** The majority of the data points are centered around the median with a few outliers. This indicates a relatively consistent range of earthquake magnitudes in the dataset.
2. **amax(g) (Peak Ground Acceleration):** There are several outliers with high values, suggesting some extreme cases of ground acceleration. However, the majority of the values are concentrated in a lower range.
3. **Liq (Liquefaction):** The 'Liq' column is binary (0 or 1), with the box plot showing a clear distinction. The plot indicates a higher count of 'Yes' cases (value 1) compared to 'No' cases (value 0), reflecting the distribution observed in the histogram.
4. **z(m) (Depth):** Depth values show a skew towards the lower range with several significant outliers, indicating some instances with notably higher depths.
5. **dw(m) (Water Table Depth):** There are numerous outliers indicating variability in water table depth, but the bulk of the data is within a lower range.
6. **svo(kPa) and s'vo(kPa) (Total and Effective Stress):** Both these columns show similar distributions with several outliers, indicating variability in stress values.
7. **qc1Ncs (Normalized Cone Penetration Resistance):** This plot shows a wide spread of data points with numerous outliers, suggesting significant variability in penetration resistance values.
8. **CRR7.5 (Cyclic Resistance Ratio):** This plot also shows a wide range with many outliers, indicating variability in the cyclic resistance ratio, which is critical for understanding soil liquefaction potential.

A histogram of the 'Liq' column illustrated the distribution of liquefaction cases, showing a significant proportion of positive instances. A correlation heatmap was generated to examine relationships between variables (Figure 2), revealing expected correlations, such as the strong relationship between total and effective stress, and uncovering a direct correlation between 'CRR7.5' and both 'qc1Ncs' and 'Vs' values. However, the anticipated strong correlation between 'CRR7.5' and 'N160cs' was not evident, suggesting the need for a deeper review of data and calculations.

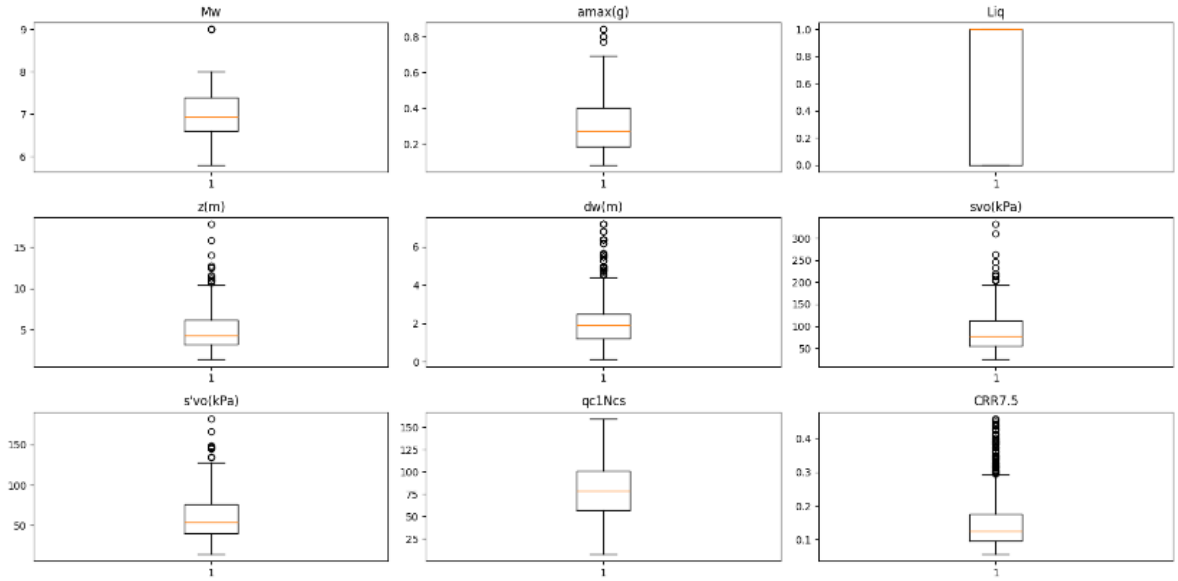



*Figure 1 box-plot for the merged CPT dataset*

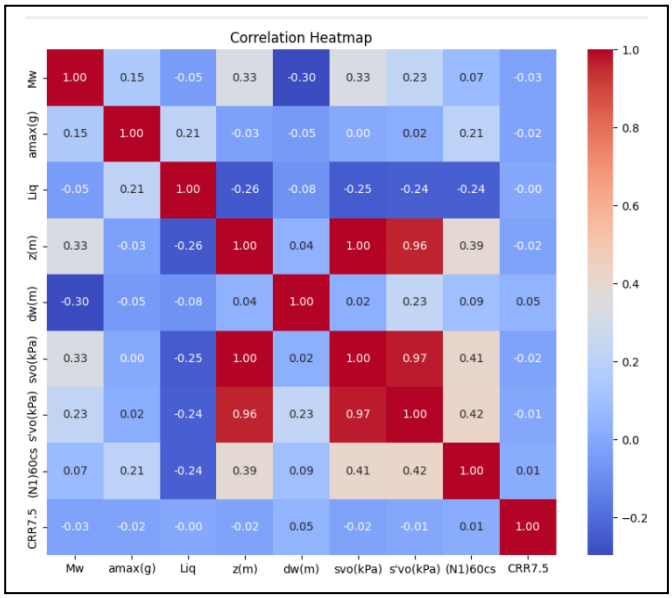
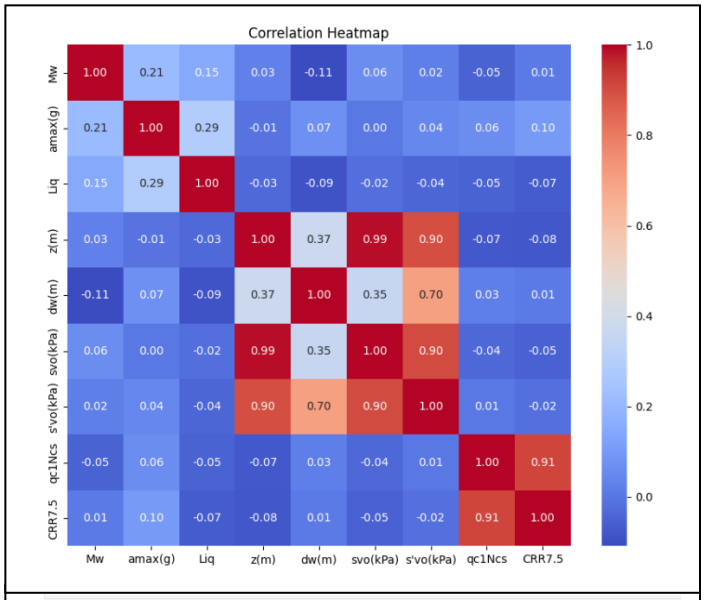
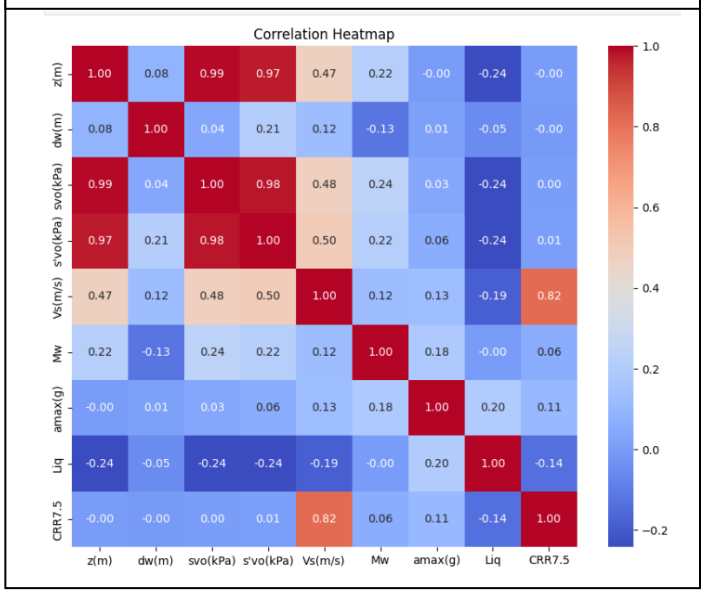
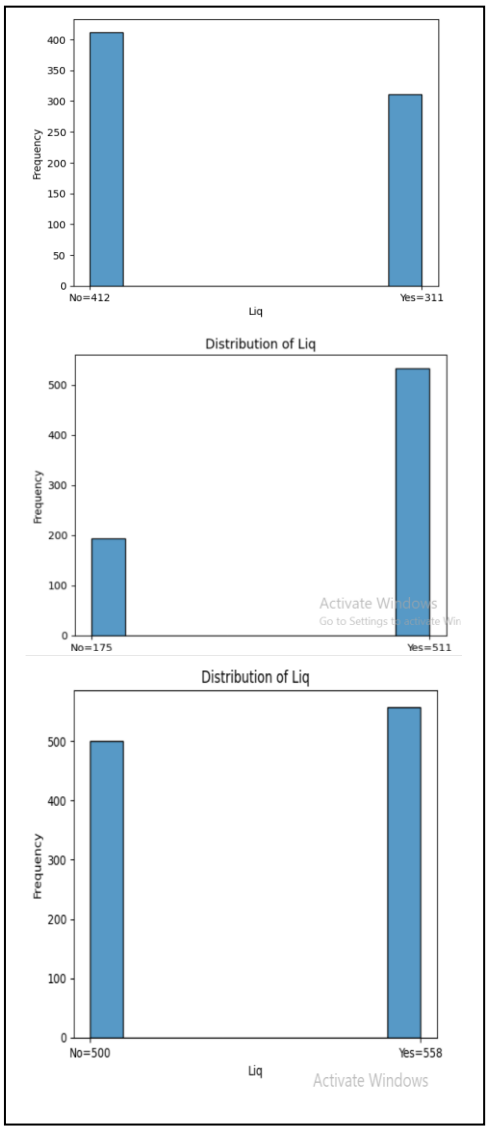



Figure2: The correlation maps and the distribution of liquefaction data sets for the merged data sets SPT, CPT and Vs respectively



## 5. Feature Engineering

The Cyclic Resistance Ratio (CRR7.5) serves as a common metric for assessing soil resistance to liquefaction under cyclic loading conditions across different soil test datasets, including Standard Penetration Test (SPT), Cone Penetration Test (CPT), and shear wave velocity (Vs). The calculation of CRR7.5 is based on the IS1893-1, 2016 standard. [8]

The rationale behind these feature engineering decisions was to harmonize data from different soil tests into a common metric, CRR7.5, enabling consistent and comparable assessment of soil liquefaction resistance.

- For the SPT dataset, CRR7.5 was calculated using the following equation: (Code 4)

$$CRR_{7.5}=\frac{1}{34-(N1)60cs}+\frac{(N1)60cs}{135}+\frac{50}{[10(N1)60cs+45]^2}-\frac{1}{200}$$

  This equation accounts for the normalized blow count values, (N1)60cs .

- For the CPT dataset, different equations were applied depending on the range of normalized cone penetration resistance ($(q_{C1N})_{CS}$ : (Code 5)

$$CRR_{7.5} = 0.833\ ((q_{C1N})_{CS}\ /\ 1000)+ 0.05 \qquad \text{for } 0 < (q_{C1N})_{CS} < 50$$

$$CRR7.5 = 93\ ((q_{C1N})_{CS}\ /\ 1000)^3+ 0.08 \qquad \text{for } 50 \leq (q_{C1N})_{CS} < 160$$

- for Vs, the CRR7.5 was calculated using the following equation: (Code 6)

$$CRR7.5 = a\left(\frac{Vs1}{100}\right)^2 + b\left(\frac{1}{Vs1*-Vs1}-\frac{1}{Vs1*}\right)$$

  where Vs1 is the normalized shear wave velocity.

[61]:
```
# Assuming 'cleaned_data' is your DataFrame
# Define the function to calculate CRR7.5 based on the given equations
def calculate_crr7_5(qc1ncs):
    if 0 < qc1ncs < 50:
        return 0.833 * (qc1ncs / 1000) + 0.05
    elif 50 <= qc1ncs < 160:
        return 93 * ((qc1ncs / 1000) ** 3) + 0.08
    else:
        print(f"Value {qc1ncs} is outside the expected range.")
        return None  # or any default value you prefer if qc1ncs is out of range

# Apply the function to the 'qc1Ncs' column to calculate 'CRR7.5'
cleaned_data['CRR7.5'] = cleaned_data['qc1Ncs'].apply(calculate_crr7_5)

# Print the DataFrame with the new column
cleaned_data
```

*Code 4 feature engineering for CPT dataset: Calculating CRR7.5 based on qc1Ncs*

[58]:

```
# Calculate the value of CRR7.5 using the given equation
merged_data_1_2_4_5['CRR7.5'] = 1 / (34 - merged_data_1_2_4_5['(N1)60cs']) + \
                                (merged_data_1_2_4_5['(N1)60cs'] / 135) + \
                                50 / (10 * merged_data_1_2_4_5['(N1)60cs'] + 45) ** 2 - \
                                1 / 200

# Print the DataFrame with the new column
merged_data_1_2_4_5
```

*Code 5 feature engineering for SPT dataset: Calculating CRR7.5 based on (N1)60cs*

]:

```
def calculate_CRR75(cleaned_data):
    a = 0.022
    b = 2.8
    Vs1_star = 200  # Given Vs1* value
    Pa = 100  # Constant value of Pa

    # Calculate Vs1 based on the provided equation
    Vs1 = cleaned_data['Vs(m/s)'] * (Pa/cleaned_data["s'vo(kPa)"])**0.25

    # Calculate CRR7.5 based on the given equation
    CRR75 = a * (Vs1/100)**2 + b * (1/(Vs1*Vs1) - 1/(Vs1_star*Vs1_star))

    # Add a new column 'CRR7.5' to cleaned_data
    cleaned_data['CRR7.5'] = CRR75
```

*Code 6 feature engineering forVs dataset: Calculating CRR7.5 based on Vs (m/sec)*

## 6. Data Transformation

In the data preprocessing phase for modeling liquidity risk (Liq), several steps were undertaken to prepare the dataset. Initially, the features and target variable were separated from the cleaned dataset, with 'Liq' being the target variable .Following this, the data was split into training, validation (dev), and testing sets using an 80-10-10 ratio (Code 7). To ensure consistency and mitigate the influence of different scales among features, standardization was applied using the `StandardScaler` from scikit-learn. This scaler was fit on the training data (`X_train`) to learn the mean and standard deviation of each feature. Subsequently, the scaler was used to transform the training set (`X_train_scaled`), validation set (`X_valid_scaled`), and test set (`X_test_scaled`) accordingly. This process ensures that all features are centered around zero with unit variance, which is essential for many machine learning algorithms, particularly those based on distances or gradients, to perform optimally.

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_s
```

80]:
```
# 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 consist
X_test_scaled = scaler.transform(X_test)    # Use scaler fitted on training set for consist
```

*Code 7 Data splitting and feature scaling*

# Model Exploration

## 1. Model Selection

The selection of neural network models for our project involved testing various configurations tailored to different datasets derived from Cone Penetration Test (CPT), Standard Penetration Test (SPT), Shear Wave Velocity (Vs), and their combinations, all aimed at predicting liquefaction susceptibility (liq). For instance, models were evaluated using CPT data featuring qc1Ncs or CRR7.5, SPT data with N160cs or CRR7.5, Vs data with Vsm/s or CRR7.5, and a combined dataset incorporating CRR7.5 from CPT, SPT, and Vs data. Among these, specific configurations like model_2 and model_3 utilized traditional ReLU and sigmoid activations, with model_3 additionally incorporating L2 regularization to enhance generalization across datasets. Alternately, models such as model_12 and model_13 employed 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. Through rigorous evaluation and validation processes, these models demonstrated robust performance metrics, ensuring their suitability for informing engineering decisions in earthquake-prone regions effectively.

## 2. Model Training

The neural network models were trained using a systematic approach to ensure robust performance in predicting liquefaction susceptibility across various datasets derived from Cone Penetration Test (CPT), Standard Penetration Test (SPT), Shear Wave Velocity (Vs), and their combinations. Each model was trained with the following details:

- **Dataset Configurations**:
    - **Feature Sets**: Depending on the dataset variant (e.g., CPT with qc1Ncs, SPT with N160cs, Vs with Vsm/s, combined dataset with CRR7.5), appropriate feature sets including seismic parameters (Mw, amax(g), z(m), dw(m), svo(kPa), s'vo(kPa)) were used.
    - **Target Variable**: Liquefaction prediction (liq) served as the target variable in all cases.

- **Model Architectures**:
    - **Model Variants**: Models such as model_2, model_3, model_12, and model_13 were tested, each with specific configurations of activation functions (e.g., ReLU, LeakyReLU), output activations (e.g., sigmoid, tanh), and regularization techniques (e.g., L2 regularization).

- **Hyperparameters:**
    - **Dense layers**: Each model consists of three dense layers: an input layer and two hidden layers.

- **Neurons:** The number of units in the hidden layers varied, typically with 64 units in the first hidden layer and 32 units in the second hidden layer.
- **Output layer**: The output layer for all models had 1 unit, reflecting the binary nature of the liquefaction prediction task (sigmoid activation for probability output or tanh activation for output in the range [-1, 1]).
- **Optimizer**: All models were compiled using the 'adam' optimizer, which adapts learning rates during training for efficient gradient descent.
- **Loss Function**: 'Binary_crossentropy' was employed as the loss function, suitable for binary classification tasks where the output predicts the probability of liquefaction occurrence.
- **Batch Size and Epochs**: Training utilized a batch size of 32 and was conducted over 150 epochs. This batch size strikes a balance between computational efficiency and model stability during gradient descent optimization.

- **Regularization**:

  - **L2 Regularization**: Models like model_3 and model_13 incorporated L2 regularization (kernel_regularizer=regularizers.l2(0.001)) to penalize large weights and prevent overfitting, crucial for improving generalization across different datasets.

- **Training Process**: (refer to Code 8)

  - **Implementation**: Models were implemented using Keras with TensorFlow backend, leveraging its high-level API for building and training neural networks efficiently.
  - **Monitoring**: Training progress and validation metrics (e.g., accuracy, loss) were monitored closely to detect overfitting and optimize model parameters accordingly.

```
# Building model_2
model_2 = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

# Compile model_2
model_2.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Print model summary
model_2.summary()

# Train model_2 with validation data
history = model_2.fit(X_train_scaled, y_train, epochs=150, batch_size=32,
                      validation_data=(X_valid_scaled, y_valid), verbose=1)

# Evaluate model_2 on the training data
train_loss, train_accuracy = model_2.evaluate(X_train_scaled, y_train)
print(f'Training Accuracy: {train_accuracy:.4f}')

# Evaluate model_2 on the development/validation data
valid_loss, valid_accuracy = model_2.evaluate(X_valid_scaled, y_valid)
print(f'Dev Accuracy: {valid_accuracy:.4f}')
# Evaluate model_2 on the test data
test_loss, test_accuracy = model_2.evaluate(X_test_scaled, y_test)
print(f'Test Accuracy: {test_accuracy}')
```

*Code 8 Model 2: Relu and Sigmoid Activation*

## 3. Model Evaluation

In evaluating the model's performance, a comprehensive set of metrics was employed across the training, validation, and test sets. Initially, confusion matrices were computed to visualize the distribution of predicted versus actual outcomes for each dataset. These matrices provide insights into the model's ability to correctly classify instances of liquefaction susceptibility (liq). Metrics such as accuracy, precision, recall, F1 score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) were calculated to quantify performance across different evaluation criteria. Accuracy measures the overall correctness of predictions, while precision and recall focus on the model's ability to correctly identify positive instances (liq) and avoid false positives, respectively. F1 score balances precision and recall, offering a single metric to gauge the model's effectiveness. Additionally, AUC-ROC evaluates the model's discriminatory ability across different threshold settings, providing a holistic view of its predictive power. These evaluations ensure 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.The evaluation merics of model_2 is shown in Table 1.

*Table 1 evaluation metrics for model_2*

| 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 |

## 4. References


[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] IS-1893-Part-1-(2016), "Criteria for Earthquake resistant design of structures, Part 1:General Provisions and buildings," *Bur. Indian Stand. New Delhi*, vol. 1893, no. December, pp. 1–44, 2016.