**Internship Report**

**Name:** Velchuri Vishnupriya  
**Internship Title:** Prediction of Compressive Strength of Concrete using Machine Learning Models  
**University:** SASTRA Deemed University

**Title:**

**Predicting Compressive Strength of Concrete Using Machine Learning Models**

**Introduction:**

Concrete strength prediction is one of the key aspects of structural design in civil engineering. The traditional methods require exhaustive lab testing and trial-and-error mix design. This could be tedious process for the large data sets.   
In this internship, I explored how machine learning (ML) can be used to simplify this and I went through the research papers that explored this concept. From the understandings of research paper, I focused on how to improve the existing results to bring more accuracy in the models. In the project specifically geopolymer concrete based data set was used, geopolymer concrete is an eco-friendly alternative that replaces traditional cement with fly ash and other industrial by-products.

**Dataset:**

* **Source**: [UCI Concrete Dataset](https://raw.githubusercontent.com/Velchuri-Vishnupriya/NIT-Trichy-Internship/main/Concrete_Data.xls)   
   Sample data sets (For training the models)
* **Total Records**: ∼1030
* **Input Features**:
  + Cement
  + Slag
  + Fly Ash
  + Water
  + Superplasticizer
  + Coarse Aggregate
  + Fine Aggregate
  + Age (in days)
* **Target Feature**: Compressive Strength (in MPa)

**Methodology:**

**Tools Used:**

* **Google Colab –** Cloud-based IDE for all coding and experiments
* **Python Libraries:**• pandas, numpy – For data preprocessing  
  • matplotlib, seaborn – For visualization  
  • scikit-learn – For Linear Regression & Random Forest  
  • tensorflow.keras – For building Artificial Neural Network (ANN)

**Preprocessing Steps:**

* Loaded dataset and handled missing data
* Normalized numerical features using MinMaxScaler
* Split data into 70% training and 30% testing

**Models Implemented:**

1. **Linear Regression:**
   * Simple baseline model assuming a linear relationship between input features and output.
   * **R² Score:** ~0.50 (Moderate fit, cannot capture complex patterns in data).
   * **CODE:**

|  |
| --- |
| from google.colab import files  uploaded = files.upload()  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import MinMaxScaler  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import r2\_score  # Load dataset  df = pd.read\_csv("Concrete\_Data.csv")  # Features and target  X = df.drop('Concrete compressive strength(MPa, megapascals) ', axis=1)  y = df['Concrete compressive strength(MPa, megapascals) ']  # Normalize features  scaler = MinMaxScaler()  X\_scaled = scaler.fit\_transform(X)  # Train-test split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.3, random\_state=42)  # Train Linear Regression model  lr\_model = LinearRegression()  lr\_model.fit(X\_train, y\_train)  # Predictions and R2 Score  lr\_preds = lr\_model.predict(X\_test)  lr\_r2 = r2\_score(y\_test, lr\_preds)  print(f"Linear Regression R² Score: {lr\_r2:.4f}")  # Plot  plt.figure(figsize=(6,4))  plt.scatter(y\_test, lr\_preds, alpha=0.6, color='blue')  plt.xlabel("Actual Compressive Strength")  plt.ylabel("Predicted Strength (Linear)")  plt.title("Linear Regression: Actual vs Predicted")  plt.grid(True)  plt.show() |
| **GRAPH:** |

1. **Random Forest Regression:**
   * Learning model based on multiple decision trees.
   * Effectively works with non-linear data set.
   * **R² Score:** ~0.88 (Strong performance and better generalization).
   * **CODE:**

|  |
| --- |
| # Step 1: Import necessary libraries  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.ensemble import RandomForestRegressor  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import r2\_score, mean\_squared\_error  # Step 2: Load dataset  df = pd.read\_csv("Concrete\_Data.csv")  # Step 3: Split into features and target  X = df.drop(columns=["Concrete compressive strength(MPa, megapascals) "])  y = df["Concrete compressive strength(MPa, megapascals) "]  # Step 4: Split into training and testing sets (70/30)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  # Step 5: Create and train the Random Forest model  rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)  rf\_model.fit(X\_train, y\_train)  # Step 6: Predict and evaluate  predictions = rf\_model.predict(X\_test)  r2 = r2\_score(y\_test, predictions)  print(f"Random Forest R² Score: {r2:.4f}")  # Step 7: Plot Actual vs Predicted  plt.scatter(y\_test, predictions, color='green', alpha=0.6)  plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')  plt.xlabel("Actual Strength (MPa)")  plt.ylabel("Predicted Strength (MPa)")  plt.title("Random Forest: Predicted vs Actual Strength")  plt.grid()  plt.show() |
| **GRAPH:** |

1. **Artificial Neural Network (ANN):**
   * Multi-layer feedforward network with:
     + **3 hidden layers**, ReLU activation
     + **1 output neuron** for regression
   * **R² Score:** ~0.88 (Best performance, on par with Random Forest)
   * **CODE:**

|  |
| --- |
| 1. # Step 1: Import libraries 2. import pandas as pd 3. import numpy as np 4. import matplotlib.pyplot as plt 5. from sklearn.model\_selection import train\_test\_split 6. from sklearn.preprocessing import StandardScaler, MinMaxScaler 7. from tensorflow.keras.models import Sequential 8. from tensorflow.keras.layers import Dense 9. # Step 2: Load dataset 10. df = pd.read\_csv("Concrete\_Data.csv") 11. # Rename columns for simplicity 12. df.columns = ['Cement', 'BlastFurnaceSlag', 'FlyAsh', 'Water', 'Superplasticizer', 13. 'CoarseAggregate', 'FineAggregate', 'Age', 'CompressiveStrength'] 14. # Step 3: Split into input (X) and output (y) 15. X = df.drop('CompressiveStrength', axis=1) 16. y = df['CompressiveStrength'] 17. # Step 4: Train-test split 18. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) 19. # Step 5: Normalize input features 20. scaler = StandardScaler() 21. X\_train = scaler.fit\_transform(X\_train) 22. X\_test = scaler.transform(X\_test) 23. # Step 6: Scale target (output) values 24. yscaler = MinMaxScaler() 25. y\_train\_scaled = yscaler.fit\_transform(y\_train.values.reshape(-1, 1)) 26. y\_test\_scaled = yscaler.transform(y\_test.values.reshape(-1, 1)) 27. # Step 7: Build ANN model 28. model = Sequential() 29. model.add(Dense(128, input\_dim=X\_train.shape[1], activation='relu')) 30. model.add(Dense(64, activation='relu')) 31. model.add(Dense(32, activation='relu')) 32. model.add(Dense(1)) # Output layer 33. # Step 8: Compile and train 34. model.compile(optimizer='adam', loss='mse', metrics=['mae']) 35. history = model.fit(X\_train, y\_train\_scaled, validation\_split=0.2, epochs=300, verbose=1) 36. # Step 9: Plot training and validation loss 37. plt.plot(history.history['loss'], label='Training Loss') 38. plt.plot(history.history['val\_loss'], label='Validation Loss') 39. plt.xlabel("Epochs") 40. plt.ylabel("Loss (MSE)") 41. plt.title("ANN Training & Validation Loss") 42. plt.legend() 43. plt.grid() 44. plt.show() 45. # Step 10: Predict on test data and inverse scale 46. scaled\_predictions = model.predict(X\_test) 47. predictions = yscaler.inverse\_transform(scaled\_predictions) |
| **GRAPH**: |

**Interpretation:**

* Both **Random Forest** and **ANN** performed very well, capturing the complex nonlinear patterns in the data.
* **Linear Regression** was not sufficient due to its simplicity and as the data was non – linearly distributed linear regression didn’t hold well.

**Evaluation Metric:**

* **R² Score (Coefficient of Determination)**: Indicates how well the predicted values approximate the actual values.
  + R² closer to 1 means better prediction.
  + ANN and Random Forest provided the most accurate predictions.

**Result Summary:**

| **Model** | **R² Score** |
| --- | --- |
| Linear Regression | 0.50+ |
| Random Forest | 0.85+ |
| ANN (Keras + Python) | 0.85+ |

**Graphical Insights:**

**ANN Loss Curve:**  
A steadily decreasing loss over epochs, confirming successful learning and convergence.

**Predicted vs Actual Plots:**

* **ANN and Random Forest** models both showed high correlation with the actual values, with most predictions falling close to the ideal line.
* **Linear Regression** displayed noticeable deviation from actual values, confirming underfitting.

**Conclusion:**

Through the internship I was able to walk through the machine learning workflows and model building with real-world data. I gained hands-on experience in data preprocessing, model training, evaluation, and result interpretation. Among the models which I tried and tested, ANN and Random Forests gave comparatively best results.

**Skills Gained:**

* Basics of Machine Learning
* Working with Python libraries like Pandas, NumPy, Keras
* Model training and evaluation using R²

*Thank you for this opportunity.*