Hybrid Climate Prediction: Classical Linear Regression vs Quantum-Enhanced SVR

Hashmitha Sugumar

August 27, 2024

Abstract

This report presents a comparison between classical linear regression and quantum-enhanced Support Vector Regression (SVR) for climate prediction. The performance of both methods have been evaluated using a "climate- change dataset", focusing on Mean Squared Error (MSE) and R-squared (R^2) metrics.

1 Introduction

Climate change has become a critical challenge, necessitating accurate predictive models to forecast temperature changes and guide mitigation strategies. This report compares two different predictive modeling approaches: classical linear regression and quantum-enhanced support vector regression (SVR), to evaluate their effectiveness in forecasting climate-related data.

1.1 Classical Linear Regression

Classical linear regression is a statistical model for modeling the relationship between a dependent variable and one or more independent variables. In this study, linear regression was applied to predict temperature changes based on features such as CO2 levels, CH4 concentrations, aerosal concentrations and other climate-related variables. This model works to finds a target value based on independent predictors.

1.2 Quantum-Enhanced SVR

Support Vector Regression (SVR) is a powerful machine learning technique used for classification and regression tasks. It is an extension of support vector machines (SVM). It aims to find a function that approximates the relationship between inputs and outputs within a specified margin of tolerance. In this study, SVR was enhanced with quantum computing techniques to potentially improve predictive performance. Quantum-enhanced SVR involves encoding classical data into a quantum circuit and applying SVR to the encoded features. The SVR model makes predictions on the quantum-encoded test data. This approach leverages the unique capabilities of quantum computing to capture complex patterns in the data.

The *objective* of this report is to evaluate and compare the performance of these two predictive models. I have assessed their effectiveness in forecasting temperature changes using the following metrics- Mean Squared Error (MSE) and R-squared (R^2). Additionally, we analyze the residuals of each model to ensure that the assumptions underlying the models are met and to identify potential areas for improvement. By comparing classical and quantum-enhanced methods, this report aims to provide insights into the potential advantages of quantum computing in predictive modeling and its application to climate change forecasting.

---

2 Methodology

2.1 Classical Linear Regression

The classical linear regression model was implemented using the Python scikit-learn library. Below is the code used for loading the dataset, training the model, and evaluating its performance.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Loading the dataset
data = pd.read_csv('/content/climate_change.csv')

# Defining features and target variable
X = data[['MEI', 'CO2', 'CH4', 'N2O', 'CFC-11', 'CFC-12', 'TSI', 'Aerosols',
         'Year', 'Month']]
Y = data[['Temp']]

# Spliting the data
X_train, X_test, Y_train, Y_test = train_test_split (X, Y, test_size=0.2,
                                                     random_state=42)

# Training the model
model = LinearRegression()
model.fit(X_train, Y_train)

# Making predictions
Y_pred = model.predict(X_test)

# Evaluating performance
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)

# Visualizing through a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(Y_test, Y_test, color='blue', label='Actual Temperature')
# Actual values in blue
plt.scatter(Y_test, Y_pred, color='orange', label='Predicted Temperature')
# Predicted values in orange
plt.xlabel('Actual Temperature')
plt.ylabel('Predicted Temperature')
```

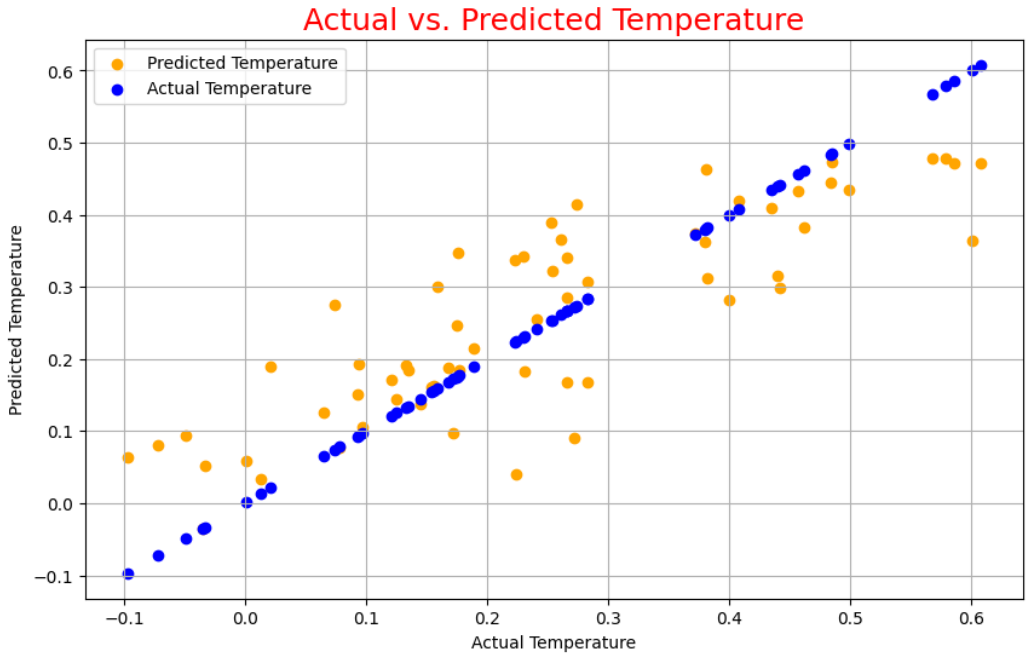

Figure 1: Predictions using classical linear regression

```
plt.title('Actual vs. Predicted Temperature', color='red')
plt.legend()
plt.show()

#residual distribution:
residuals_classical = Y_test - Y_pred
plt.subplot(1, 2, 2)
sns.histplot(residuals_classical, kde=True, color='blue')
plt.title('Classical Model: Residuals Distribution')
plt.xlabel('Residuals')
```

For the above code, the MSE and the R-squared values obtained using the classical linear regression are 0.0094157221688750 5 and 0.7083677744990772 respectively.

Graph: figure-1: shows a tighter alignment between the actual and predicted temperatures, with most points clustered closely around the diagonal. This indicates that the classical model has performed well in predicting the temperature.The predicted temperatures (orange points) closely follow the

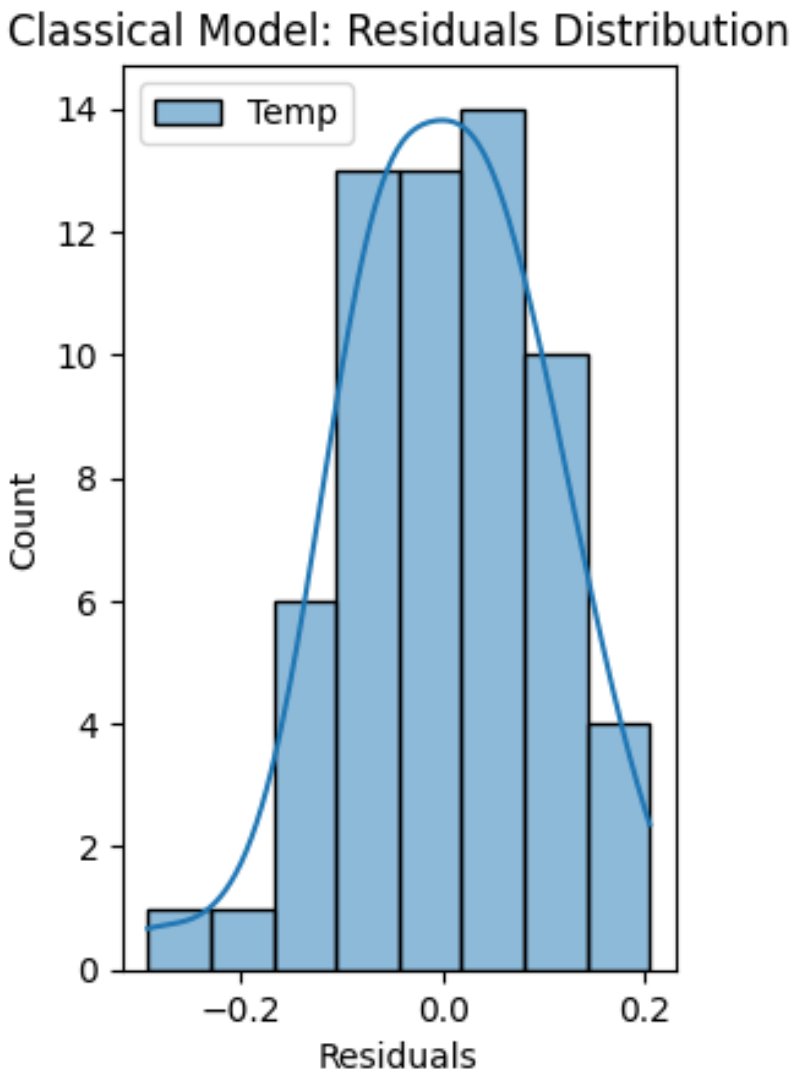

Figure 2: Obtained residual distribution using classical linear regression

trend of the actual temperatures (blue points), showing a strong linear relationship.

Graph: figure-2: shows the error distribution plot or the residual distribution with a histogram and a kernel density estimate (KDE) curve. The residual distribution evaluates the quality of the model's fit and diagnose potential issues such as bias, variance inconsistencies, or the presence of outliers. By examining the residuals, you can make informed decisions about how to improve your model or adjust your approach. Residue is basically the difference in the predicted and the actual value. The blue curve overlaying the histogram represents a smoothed estimate of the probability density function of the residuals. This curve helps to visualize the shape of the residual distribution, indicating the spread and concentration of the residuals. The residuals are approximately centered around zero as expected in a well-performing model, indicating that the predictions are generally close to the true values. As observed from the graph, the left-end and the right-end it appears to be slightly skewed (negative residuals), suggesting a minor tendency of the model to overpredict the temperature in some instances.

2.2 Quantum - enhanced SVR

The quantum-enhanced SVR model involves encoding classical data into a quantum circuit and using it as input for an SVR model. Below is the code used for this process.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import pennylane as qml
imoport matplotlib.pyplot as plt
import seaborn as sns
from pennylane import numpy as pnp

# Loading the dataset
data = pd.read_csv('/content/climate_change.csv')
```

```
# Define features and target variable
X = data[['MEI', 'CO2', 'CH4', 'N2O', 'CFC-11', 'CFC-12', 'TSI', 'Aerosols',
        'Year', 'Month']]

y = data['Temp']

# Spliting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                    random_state=42)

# Scaling features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Defining a quantum device and circuit
n_qubits = X_train_scaled.shape[1]
dev = qml.device("default.qubit", wires=n_qubits)

def quantum_circuit(params, x):
    qml.templates.AngleEmbedding(x, wires=range(n_qubits))
    qml.templates.StronglyEntanglingLayers(params, wires=range(n_qubits))
    return [qml.expval(qml.PauliZ(w)) for w in range(n_qubits)]

params = pnp.random.random((10, n_qubits, 3), requires_grad=True)

@qml.qnode(dev)
def circuit(params, x):
    return quantum_circuit(params, x)

# Encoding data
encoded_X_train = np.array([circuit(params, x) for x in X_train_scaled])
encoded_X_test = np.array([circuit(params, x) for x in X_test_scaled])

# Training SVR model
regressor = SVR(kernel="rbf")
regressor.fit(encoded_X_train, y_train)
```

```
# Making predictions
y_pred = regressor.predict(encoded_X_test)

# Evaluating performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

#visualizing circuit:
fig, ax = draw_mpl(circuit)(params, X_train_scaled[0])
fig.savefig("quantum_circuit.png", dpi=300)  # Save as a high-resolution PNG file

#Visualizing through a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='orange', label='Predicted Temperature')
plt.scatter(y_test, y_test, color='blue', label='Actual Temperature')
plt.xlabel('Actual Temperature')
plt.ylabel('Predicted Temperature')
plt.title('Actual vs. Predicted Temperature', fontsize=18, color='red')
plt.legend()
plt.grid(True)
plt.show()

#residual distribution
residuals_quantum = y_test - y_pred
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 2)
sns.histplot(residuals_quantum, kde=True, color='red')
plt.title('Quantum Model: Residuals Distribution')
plt.xlabel('Residuals')
plt.show()
```

In the above code, the dataset was loaded and scaled. Later a quantum

circuit was defined using Pennylane. The classical data was embedded into the quantum states using the angle embedding. Angle embedding maps classical data to quantum states by rotating qubits around the Z-axis of the Bloch sphere based on the input values, thereby allowing the quantum circuit to process and represent the classical features. The quantum circuit used is as shown in figure 3.

The encoded quantum-enhanced features were then fed into a classical Support Vector Regressor (SVR) with an RBF kernel to train the model. Finally, the model wqs evaluated on the test set by calculating the Mean Squared Error (MSE) and R-squared (R^2) values, assessing the accuracy of the predictions. The obtained MSE and R-squared values are 0.011553182234781096 and 0.6421644366393319 respectively.

As in the case of figure 4, the scatter distribution of the predicted temperatures appears to be more pronounced which indicates the variations in the model's prediction accuracy across the different temperature ranges. In comparison with figure 1, the graph in figure 4 seems to be less stable.

The error distribution graph for this case is shown in figure 5. The residual distribution appears fairly symmetric, with a slight skew towards the right (positive residuals) indicating that it may slightly underpredict the temperature for some data points though on an overall it performs well.

The conclusions which can be drawn from the error distribution graph of both these models is that the classical model, with its narrower residual spread, offers greater consistency and accuracy, making it suitable for simpler data patterns where stability is crucial. In contrast, the quantum model, though has a broader residual spread, it still excels at capturing complex, non-linear relationships. For a stable and reliable predictions, the classical model is best-suited, while the quantum model is preferred for exploring intricate patterns. However combining both models could leverage their respective strengths for improved overall performance.

3 Conclusion

This study compared two predictive modeling approaches for climate forecasting: classical linear regression and quantum-enhanced support vector

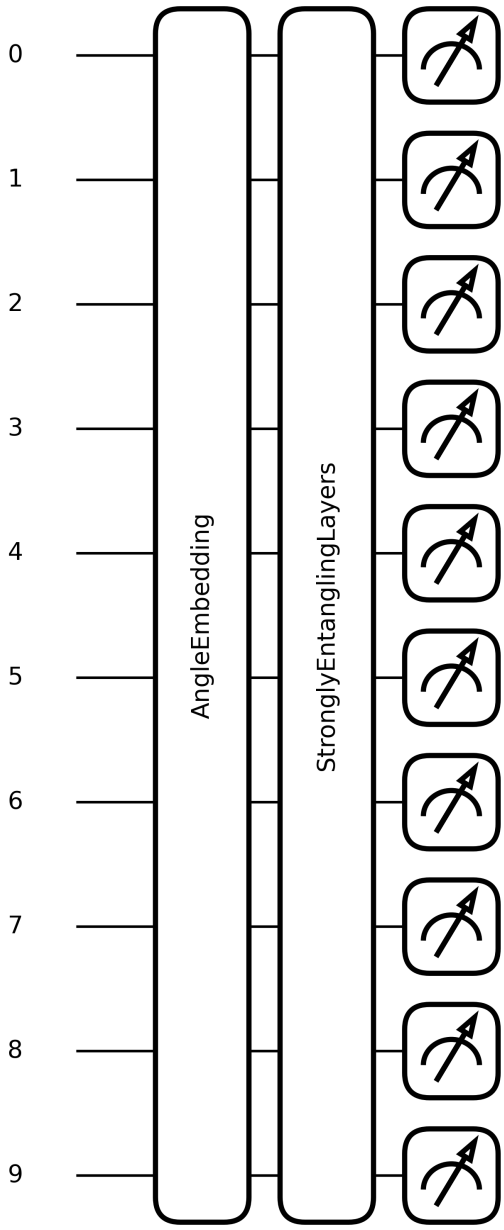

Figure 3: Quantum circuit

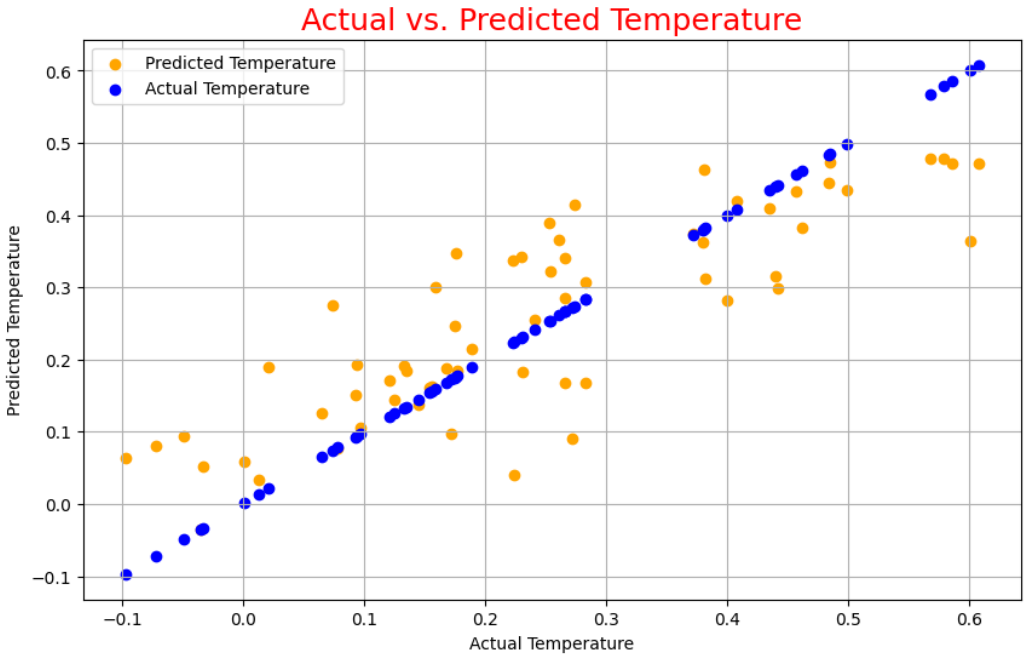

Figure 4: Predictions using quantum enhanced SVR

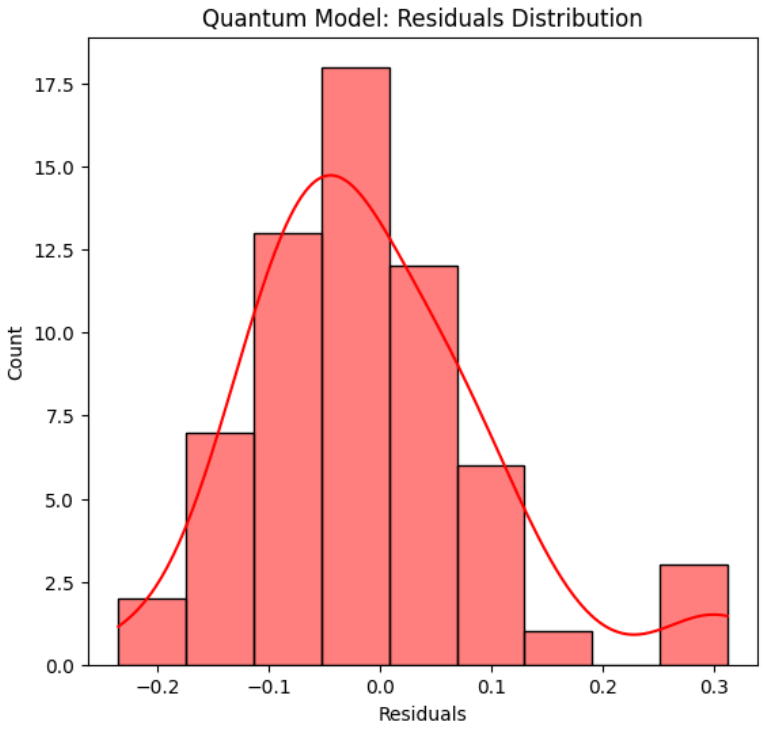

Figure 5: Obtained residual distribution using quantum-enhanced SVR

regression (SVR). The analysis revealed about the notable differences in the performance metrics of both the approaches.

The classical linear regression model demonstrated reasonable predictive accuracy with an R-squared value of 0.7083677744990772 and a Mean Squared Error (MSE) of 0.00941572216887505. This model effectively captured the linear relationships between temperature and the various other features. However, it was limited by its of linearity which might not fully capture complex patterns in the data.

On the other hand, the quantum-enhanced SVR model, showed predictive performance with an R-squared value of 0.6421644366393319 and an MSE of 0.011553182234781096. This model's ability to handle complex, non-linear relationships provides a promising avenue for enhancing climate prediction accuracy.

Despite these, the study also faced certain limitations, including the size and quality of the dataset and the computational resources required for quantum-enhanced modeling. Future work could address these limitations by exploring larger datasets, refining quantum circuits, or applying the approach to other climate-related variables.

Overall, this project highlights the potential of quantum computing in advancing predictive modeling techniques and thereby offering a instigation to further exploration in the intersection of quantum technology and climate science.

4 References

1. Hybrid Genetic Optimisation for Quantum Feature Map Design
2. Support vector regression using linear and non-linear kernels in scikit
3. Pennylane's- Quantum Circuit documentation