# Comparative Analysis of Regression Models

## 1. Linear Regression (Simple and Multiple)

**Mathematical Principles:**

* **Simple Linear Regression** models the relationship between two variables using the equation  y=β0+β1x+*ϵ*, where *y* is the dependent variable, *x* is the independent variable, β0​ is the intercept, β1​ is the slope, and ϵ is the error term.
* **Multiple Linear Regression** extends this to multiple predictors: y=β0+β1x1+β2x2+...+βnxn+ϵ
* **Assumptions and Limitations:**
* Assumes linearity, independence of errors, homoscedasticity (constant variance of errors), and normality of error terms.
* Sensitive to outliers and multicollinearity among predictors.

**Advantages:**

* Simple to implement and interpret.
* Computationally efficient.

**Disadvantages:**

* Limited to linear relationships; poor performance with non-linear data.

**Applications:**

* Suitable for datasets where relationships are expected to be linear, such as sales forecasting based on advertising spend.

## 2. Polynomial Regression

**Mathematical Principles:**

* Extends linear regression by adding polynomial terms:

 y=β0+β1x+β2x2+...+βnxn+ϵ*y*=*β*0​+*β*1​*x*+*β*2​*x*2+...+*βn*​*xn*+*ϵ*.

**Assumptions and Limitations:**

* Similar assumptions as linear regression but can fit non-linear relationships.
* Risk of overfitting with high-degree polynomials.

**Advantages:**

* Can model non-linear relationships effectively.

**Disadvantages:**

* More complex interpretation; prone to overfitting if not managed properly.

**Applications:**

* Useful in scenarios where data follows a curvilinear trend, such as growth curves.

## 3. Support Vector Regression (SVR)

**Mathematical Principles:**

* SVR uses a hyperplane to separate data points while maximizing the margin around it. It aims to find a function that deviates from actual observed values by a value no greater than a specified threshold.

**Assumptions and Limitations:**

* Assumes that data can be separated by a hyperplane in high-dimensional space.
* Sensitive to choice of kernel and parameters.

**Advantages:**

* Effective in high-dimensional spaces; robust against overfitting in high dimensions.

**Disadvantages:**

* Computationally intensive; requires careful tuning of parameters.

**Applications:**

* Suitable for complex datasets with non-linear relationships, such as financial forecasting.

## 4. Decision Tree Regression

**Mathematical Principles:**

* Uses a tree-like model of decisions to predict outcomes based on input features. Splits data into subsets based on feature values.

**Assumptions and Limitations:**

* Assumes that the relationship between features and target can be captured by recursive partitioning.
* Prone to overfitting if not pruned properly.

**Advantages:**

* Easy to interpret; handles both numerical and categorical data well.

**Disadvantages:**

* Can create overly complex trees leading to poor generalization.

**Applications:**

* Useful in scenarios requiring interpretable models, such as customer segmentation.

## 5. Random Forest Regression

**Mathematical Principles:**

* An ensemble method that constructs multiple decision trees during training and outputs the mean prediction of individual trees for regression tasks.

**Assumptions and Limitations:**

* Assumes that individual decision trees can capture different aspects of the data.
* Requires more computational resources than single decision trees.

**Advantages:**

* Reduces overfitting compared to individual decision trees; robust against noise.

**Disadvantages:**

* Less interpretable than single decision trees; longer training time.

**Applications:**

* Effective for large datasets with many features, such as predicting housing prices based on numerous factors.

## 6. Ridge Regression

**Mathematical Principles:**

* A type of linear regression that includes a penalty term to control for multicollinearity among predictors:

*L*=∣∣*y*−*Xβ*∣∣2+*λ*∣∣*β*∣∣2

where L is the loss function, X is the feature matrix, y is the target vector, β are coefficients, and λ is the regularization parameter.

**Assumptions and Limitations:**

* Assumes linearity; does not perform variable selection (all predictors are retained).

**Advantages:**

* Handles multicollinearity well; improves model generalization.

**Disadvantages:**

* Coefficients are biased towards zero; may not perform well if true relationships are non-linear.

**Applications:**

* Suitable for datasets with many correlated predictors, such as genetics studies.

## 7. Lasso Regression

**Mathematical Principles:**

* Similar to ridge regression but uses L1 regularization:

*L*=∣∣*y*−*Xβ*∣∣2+*λ*∣∣*β*∣∣1​  
This encourages sparsity in coefficients (some coefficients may become exactly zero).

**Assumptions and Limitations:**

* Assumes linearity; can select variables by forcing some coefficients to zero.

**Advantages:**

* Performs variable selection; useful when dealing with many predictors.

**Disadvantages:**

* Can be unstable when predictors are highly correlated; sensitive to choice of regularization parameter.

**Applications:**

* Effective for high-dimensional datasets where feature selection is crucial, such as text classification tasks.

## Comparative Summary Table

| **Model** | **Mathematical Principle** | **Assumptions** | **Advantages** | **Disadvantages** | **Suitable Applications** |
| --- | --- | --- | --- | --- | --- |
| Linear Regression | y=β0+β1x+ϵ*y*=*β*0​+*β*1​*x*+*ϵ* | Linearity | Simple interpretation | Sensitive to outliers | Sales forecasting |
| Polynomial Regression | y=β0+∑i=1nβixi+ϵ*y*=*β*0​+∑*i*=1*n*​*βi*​*xi*+*ϵ* | Non-linearity | Models non-linear relationships | Risk of overfitting | Growth curves |
| Support Vector Regression | Hyperplane maximization | Data separability | Effective in high dimensions | Computationally intensive | Financial forecasting |
| Decision Tree Regression | Recursive partitioning | Recursive splits | Easy interpretation | Prone to overfitting | Customer segmentation |
| Random Forest Regression | Ensemble of decision trees | Multiple trees | Robust against noise | Less interpretable | Predicting housing prices |
| Ridge Regression | Regularized linear regression | Linearity | Handles multicollinearity | Biased coefficients | Genetics studies |
| Lasso Regression | L1 regularized linear regression | Linearity | Variable selection | Unstable with correlated predictors | Text classification |

**Explanation of Chosen Metrics**

1. **Mean Squared Error (MSE)**:
   * **Definition**: MSE is the average of the squared differences between predicted and actual values, mathematically expressed as:
   * MSE=1/n∑(yi−y^i)^2
   * where yi​ is the actual value, y^i​ is the predicted value, and *n* is the number of observations.
   * **Appropriateness**: MSE penalizes larger errors more than smaller ones due to squaring, making it sensitive to outliers. This property is useful when large errors are particularly undesirable in applications like financial forecasting or risk assessment.
2. **R-squared (R2*R*2)**:
   * **Definition**: R2*R*2 measures the proportion of variance in the dependent variable that can be explained by the independent variables in the model. It is calculated as:
   * R2=1−RSS/TSS
   * ​where RSS is the residual sum of squares and TSS is the total sum of squares.
   * **Appropriateness**: An R2 value of 1 indicates perfect prediction, while a value of 0 suggests that the model does not explain any variability in the target variable. It provides insight into how well the model captures the underlying data structure.

**Hyperparameter Tuning Strategy**

Hyperparameter tuning is crucial for optimizing model performance, particularly for models like Support Vector Regression (SVR), Random Forest, Ridge, and Lasso regression. The strategy involves systematically searching through a specified subset of hyperparameters to find the combination that yields the best model performance.

Techniques Used:

* **GridSearchCV**:
  + This method exhaustively considers all parameter combinations specified in a grid. It evaluates each combination using cross-validation to ensure that results are robust and not specific to a single train-test split.
  + For example, with Random Forest, we might tune parameters such as n\_estimators (number of trees) and max\_depth (maximum depth of trees).
* **RandomizedSearchCV**:
  + This method samples a fixed number of parameter settings from specified distributions. It is more efficient than GridSearchCV when dealing with a large number of hyperparameters or when computational resources are limited.
  + For instance, tuning SVR might involve sampling different values for C (regularization parameter) and epsilon (tube width).

This process ensures that we select hyperparameters that lead to improved predictive performance while mitigating overfitting risks.

## In the current test conducted Random Forest served as the best model because of the lowest MSE value and highest R^2 value.

# Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed primarily for processing data with a grid-like topology, such as images. Their architecture is tailored to automatically and adaptively learn spatial hierarchies of features from input images, making them highly effective for tasks like image classification, object detection, and more.

Key Components of CNNs

1. **Convolutional Layers**:
   * **Function**: The convolutional layer is the core building block of a CNN. It applies a set of learnable filters (kernels) to the input image to produce feature maps. Each filter is designed to detect specific features such as edges, textures, or shapes.
   * **Operation**: The filter slides over the input image and performs an element-wise multiplication followed by summation, producing a feature map that highlights the presence of specific features.
   * **Output**: The output of this layer is a set of feature maps that represent different aspects of the input data.
2. **Pooling Layers**:
   * **Function**: Pooling layers reduce the spatial dimensions (width and height) of the feature maps while retaining the most important information. This helps decrease computational load and mitigates the risk of overfitting.
   * **Types**:
     + **Max Pooling**: Takes the maximum value from each patch of the feature map.
     + **Average Pooling**: Computes the average value from each patch.
   * **Output**: The output is a downsampled version of the feature maps, which retains essential features while reducing dimensionality.
3. **Activation Functions**:
   * **Function**: Activation functions introduce non-linearity into the model, allowing it to learn complex patterns. Without non-linearity, a CNN would behave like a linear model, limiting its ability to capture intricate relationships in data.
   * **Common Functions**:
     + **ReLU (Rectified Linear Unit)**: Applies a threshold at zero; all negative values become zero while positive values remain unchanged. This function helps in speeding up convergence during training.
     + Other functions like Sigmoid or Tanh can also be used but are less common in deeper networks due to issues like vanishing gradients.
4. **Fully Connected Layers**:
   * After several convolutional and pooling layers, the high-level reasoning in the neural network is performed via fully connected layers. These layers connect every neuron in one layer to every neuron in the next layer.
   * The output from the final pooling layer is flattened into a vector and passed through these fully connected layers for classification.
5. **Output Layer**:
   * The final layer typically uses a Softmax activation function for multi-class classification tasks, providing probabilities for each class based on the learned features.

**Explanation of Code**

* **Data Loading and Preprocessing**: The MNIST dataset is loaded and reshaped to include a channel dimension since CNNs expect input in a specific format. The pixel values are normalized to be between 0 and 1.
* **Model Architecture**:
  + A sequential model is built with two convolutional layers followed by max pooling layers.
  + The output from these layers is flattened before being passed through fully connected layers.
* **Compilation and Training**: The model is compiled with the Adam optimizer and categorical crossentropy loss function suitable for multi-class classification. It is then trained on the training data with validation on test data.
* **Evaluation and Visualization**: Finally, the model's performance is evaluated on test data and training history is visualized.

This simple CNN architecture effectively demonstrates how convolutional networks operate for image classification tasks while following best practices in coding style with proper comments and structure.

# 3. Long Short-Term Memory Networks (LSTMs)

Architecture and Key Components

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to overcome the limitations of traditional RNNs, particularly the vanishing gradient problem. LSTMs are capable of learning long-term dependencies in sequential data, making them suitable for tasks like time series forecasting and natural language processing.

1. **Memory Cell**

* Each LSTM unit contains a **memory cell**, which is responsible for maintaining information over time. The memory cell has an internal state that can be updated, allowing it to store information for long periods.

2. **Gates**

LSTMs utilize three types of gates to control the flow of information into and out of the memory cell:

* **Input Gate**:
  + This gate determines how much of the new input should be added to the memory cell. It takes the current input and the previous hidden state as inputs and applies a sigmoid activation function to produce values between 0 and 1, indicating how much information to let through.
* **Forget Gate**:
  + The forget gate decides what information from the memory cell should be discarded. Similar to the input gate, it uses a sigmoid activation function to output values between 0 and 1, where values closer to 0 mean "forget" and values closer to 1 mean "keep."
* **Output Gate**:
  + The output gate controls what part of the memory cell's internal state should be output to the next layer or time step. It also uses a sigmoid activation function and works in conjunction with a tanh activation applied to the memory cell state.

3. **Internal State Update**

The internal state of the LSTM is updated based on the input gate, forget gate, and new candidate values generated from the current input and previous hidden state. This allows LSTMs to maintain relevant information over long sequences while discarding irrelevant data.

Explanation of Code

1. **Imports**: The necessary libraries are imported at the beginning.
   * numpy and pandas for numerical operations and data manipulation.
   * matplotlib.pyplot for plotting results.
   * tensorflow.keras for building and training the LSTM model.
   * MinMaxScaler from sklearn for scaling data.
2. **Data Generation**: A synthetic sine wave with noise is generated as a sample dataset.
3. **Data Preparation**:
   * The data is reshaped into a two-dimensional array.
   * The data is scaled to a range between [0, 1] using MinMaxScaler.
4. **Creating Sequences**: A function is defined to create sequences of data suitable for LSTM input:
   * Each sequence consists of a specified number of previous timesteps (e.g., time\_step=10) used to predict the next value.
5. **Reshaping Input**: The input data is reshaped into three dimensions required by LSTMs: [samples, time steps, features].
6. **Building the Model**:
   * A Sequential model is created with two LSTM layers followed by dropout layers for regularization.
   * The final layer is a dense layer with one output neuron for regression tasks.
7. **Compiling the Model**: The model is compiled with Adam optimizer and mean squared error loss function.
8. **Training**: The model is trained on the prepared dataset for a specified number of epochs.
9. **Making Predictions**: After training, predictions are made on the same dataset.
10. **Plotting Results**: Finally, true values and predictions are plotted for visual comparison.