Final Project Deliverable: IEOR4578 Appendix

Ishita Pundir, Saumya Kothari, Tushar Bura

May 1, 2024

The **"ElectricityLoadDiagrams20112014"** dataset is a comprehensive collection of electricity consumption data recorded from 2011 to 2014. It features the energy usage details of 370 distinct points or clients, making it an invaluable resource for studies in time-series analysis within the field of computer science. This dataset lends itself well to a variety of tasks such as regression and clustering due to its detailed recording of consumption in kilowatts every 15 minutes, leading to a total of 140,256 features across all clients.

Each data point is meticulously noted without any missing values, ensuring a high level of data integrity and reliability for analysis.

Here is an appendix of the detailed explanation and math behind each of the models used throughout the Final Project:

1 K-means Clustering

K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. Here's how it works mathematically:

1. **Initialization**: Start by choosing k initial centroids. These centroids can be random points, or they might be chosen by some other method designed to speed up convergence and improve robustness.

2. **Assignment step**: Assign each observation to the nearest centroid. "Nearest" is typically defined using the Euclidean distance, although other distances can also be used. Mathematically, each data point x_i is assigned to a cluster C_j such that:

$$d(x_i, \mu_j) \leq d(x_i, \mu_{j'}) \quad \forall j' \neq j$$

   where $d(a, b)$ is the distance between points a and b, and μ_j is the centroid of cluster C_j.

3. **Update step**: After all objects have been assigned, recalculate the positions of the k centroids. The new centroid of each cluster is calculated as the mean of all points assigned to that cluster:

   $$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

   where $|C_j|$ is the number of data points in cluster C_j.

4. **Repeat**: Steps 2 and 3 are repeated until the centroids no longer move significantly, or the assignments no longer change, or a maximum number of iterations is reached.

Objective Function

The objective of K-means is typically to minimize an objective function known as the within-cluster sum of squares (WCSS). Mathematically, the objective function is:

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

where $\|x_i - \mu_j\|$ is the Euclidean distance between x_i and μ_j. The goal is to find the values of μ_j that minimize J.

K-means is a powerful clustering technique that is widely used due to its simplicity and speed, but understanding its limitations is important for practical applications.

2 Choosing the Best k in K-means Clustering Using the Silhouette Score

The Silhouette Score is a method to determine the optimal number of clusters k in K-means clustering. It measures how similar an object is to its own cluster compared to other clusters. This metric helps in assessing the appropriateness of the number of clusters used by measuring cohesion and separation.

Mathematical Definition

For each data point i, calculate the following:

1. **Average intra-cluster distance ($a(i)$)**: The average distance between i and all other data points in the same cluster.

   $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

   where C_i is the cluster of the ith point and $d(i, j)$ is the distance between the ith and jth points.

2. **Average nearest-cluster distance ($b(i)$)**: The lowest average distance of i to all points in any other cluster, of which i is not a member.

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

C_k is a cluster other than C_i.

The silhouette score $s(i)$ for each individual data point i is then given by:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

This score ranges from -1 to 1:

- A score of +1 indicates very good matching to its own cluster and poor matching to neighboring clusters.
- A score of 0 indicates that the point is close to the decision boundary between two neighboring clusters.
- A score of -1 indicates incorrect cluster assignment.

Calculating the Optimal k

To determine the best k:

1. Compute the silhouette score for each data point across different values of k.
2. For each k, average all the individual silhouette scores of the points in the dataset.
3. The optimal number of clusters k is the one that maximizes the average silhouette score, indicating good cohesion and separation.

3 Random Forest: Explanation and Mathematical Foundations

Explanation

Random Forest is an ensemble learning technique used for both regression and classification tasks. It combines multiple decision trees to improve the generalization ability of the model and reduce overfitting. Here's how it works:

1. **Bootstrap Aggregating (Bagging)**: Random subsets of the training data are selected for each tree through bootstrap sampling. Each subset can have repeated samples, and some samples may be left out.

2. **Feature Randomness**: When building each tree, the model introduces randomness by choosing the best split from a subset of features randomly chosen at that node. This ensures that the trees are diverse.
3. **Building Trees**: Each tree is grown to the largest extent possible without pruning.
4. **Majority Voting or Averaging**: For classification, the prediction by the forest is the mode of the predictions from all trees. For regression, it is the average.

Mathematical Foundations

Decision Tree Construction

Each decision tree is constructed based on recursive binary splitting:

- **Gini Impurity** (Classification): A measure aiming to minimize the probability of misclassification.

$$G = 1 - \sum_{i=1}^{n} p_i^2$$

  where p_i is the proportion of samples belonging to class i at a node.

- **Mean Squared Error** (Regression): Measures the average of the squares of the differences between observed and predicted outcomes.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

  where Y_i are the actual observations and $\hat{Y}_i$ are the predictions from the tree.

Ensemble Prediction

The final prediction in Random Forest is an aggregate of predictions from individual trees:

- **Classification**:

$$\text{Final Prediction} = \text{mode}(\{Y_1, Y_2, \ldots, Y_t\})$$

  where Y_t are predictions from each tree.

- **Regression**:

$$\text{Final Prediction} = \frac{1}{T}\sum_{t=1}^{T} Y_t$$

  where Y_t are predictions from each tree, and T is the total number of trees.

Benefits and Limitations

Benefits

- High accuracy and robustness to noise and outliers.
- Effective for large datasets and high-dimensional spaces.
- Provides feature importance scores.

Limitations

- Reduced model interpretability compared to single decision trees.
- Computationally intensive for very large datasets.
- Dependent on tuning multiple hyperparameters.

4 Explanation and Mathematical Background of Temporal Fusion Transformer (TFT)

Core Components of TFT

The Temporal Fusion Transformer (TFT) is a machine learning model designed for time-series forecasting. It incorporates several key components:

1. **Gating Mechanisms**: Similar to LSTM networks, gating mechanisms control the flow of information, allowing the model to selectively remember or forget information.
2. **Temporal Processing**: Includes components like variable selection networks, temporal self-attention mechanisms, and positional encodings to capture temporal patterns effectively.
3. **Interpretable Multi-head Attention**: Utilizes multi-head attention to provide insights into important parts of the data for predictions.
4. **Static and Dynamic Inputs**: Separates static and dynamic inputs to handle time-varying and time-invariant features separately.
5. **Quantile Output Layers**: Outputs predictions for multiple quantiles to estimate uncertainty in forecasts.

Mathematical Formulation

The TFT model involves several mathematical formulations for its components:

1. **Gating Mechanisms**:

$$\text{Gated Input} = \sigma(\mathbf{W}_g \mathbf{x}_t + \mathbf{b}_g) \odot \mathbf{x}_t$$

2. **Temporal Self-Attention**:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

3. **Quantile Output Layers**:

$$\hat{y}_\tau = \mathbf{W}_\tau \mathbf{h}_T + \mathbf{b}_\tau$$

Where σ is the sigmoid activation function, $\mathbf{W}_g$ and $\mathbf{W}_\tau$ are weight matrices, $\mathbf{b}_g$ and $\mathbf{b}_\tau$ are bias vectors, $\odot$ denotes element-wise multiplication, and $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are query, key, and value matrices respectively.

Benefits and Challenges

Benefits:

- High accuracy in forecasting.
- Interpretability through attention mechanisms.

Challenges:

- Complexity of the model architecture.
- Data requirements for effective training.

5 DeepAR: Explanation and Mathematical Foundations

DeepAR is a probabilistic forecasting algorithm developed by Amazon for time-series data. It uses a recurrent neural network (RNN) framework to predict future values by training on a large number of related time series, incorporating complex patterns and providing uncertainty estimates.

Core Components of DeepAR

1. **Autoregressive Nature**: DeepAR is autoregressive, using past data points to predict future ones, extending traditional ARIMA models with a neural network approach.
2. **RNN Architecture**: It typically utilizes LSTM or GRU layers to capture long-term dependencies in the data.
3. **Probabilistic Outputs**: DeepAR models the output distribution of future points, not just point estimates.
4. **Feature Incorporation**: It can handle both static and dynamic covariates, adding contextual understanding to the model.

Mathematical Formulation

DeepAR utilizes an RNN to process data sequences. Here's how it is structured mathematically:

1. **Network Input**: At each time step t, the network inputs the observed value x_t and outputs parameters for the future value's distribution.
2. **RNN Equations (using LSTM)**:
   - **Forget Gate**:
   $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
   - **Input Gate**:
   $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
   - **Output Gate**:
   $$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
   - **Cell State Update**:
   $$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
   - **Hidden State Update**:
   $$h_t = o_t \odot \tanh(c_t)$$

   Here, σ denotes the sigmoid activation function, $\odot$ represents element-wise multiplication, and W, b are weights and biases for the respective gates.
3. **Output Layer**: The output layer uses the hidden state h_t to parameterize the Gaussian distribution:
   $$\mu_{t+1}, \sigma^2_{t+1} = W_h \cdot h_t + b_h$$
   Predicting the mean μ_{t+1} and variance σ^2_{t+1} for the next time point.

Benefits and Limitations

Benefits:

- Can model complex nonlinear relationships in time-series data.
- Provides distribution-based forecasts, quantifying prediction uncertainty.
- Scalable to large datasets with many related time series.

Limitations:

- Requires extensive data for effective training.
- More complex and resource-intensive compared to simpler models.

6 ARIMA Model: Explanation and Mathematical Background

The ARIMA (AutoRegressive Integrated Moving Average) model is a popular statistical method for time-series forecasting, well-suited for analyzing non-stationary series which can be made stationary by differencing.

Explanation of the Components

1. **AR (AutoRegressive) part**: Predicts the current value of the series as a linear combination of previous values, capturing momentum from past values.
2. **I (Integrated) part**: Involves differencing the series to make it stationary, reducing or eliminating trends and cycles.
3. **MA (Moving Average) part**: Models the current value of the series as a linear combination of past forecast errors, capturing shocks from past errors.

Mathematical Formulation

An ARIMA model is typically expressed as ARIMA(p, d, q), where:

- p is the number of lag observations in the model (AR terms),
- d is the degree of differencing,
- q is the size of the moving average window (MA terms).

Equation of the ARIMA model:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = (1 + \sum_{j=1}^{q} \theta_j L^j)\epsilon_t$$

where:

- L is the lag operator, $LX_t = X_{t-1}$,
- ϕ_i are the parameters of the AR part,
- θ_j are the parameters of the MA part,
- ϵ_t is white noise error terms at time t,
- X_t is the time series.

Steps to Model with ARIMA

1. **Identification**: Determine p, d, and q by examining the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of the differenced series.

2. **Estimation**: Estimate the parameters ϕ and θ using Maximum Likelihood Estimation (MLE) or least squares.

3. **Diagnostic Checking**: Analyze the residuals to ensure they resemble white noise, confirming the model fit.

4. **Forecasting**: Use the model to make forecasts once it is adequately fitted.

Benefits and Limitations

Benefits:

- Flexible model capable of modeling a wide range of time series data.
- Supported by strong theoretical foundations.

Limitations:

- Parameter selection (p, d, q) can be challenging and requires expertise.
- Poor at handling seasonal effects unless modified into Seasonal ARIMA (SARIMA).
- Assumes linear relationships between lagged variables, not suitable for complex or non-linear series.

7 SARIMA Model: Explanation and Mathematical Foundations

The SARIMA (Seasonal AutoRegressive Integrated Moving Average) model extends the ARIMA model to address seasonal variations in time series data, making it suitable for forecasting series with clear pattern repetitions over calendar time.

Explanation of the Components

SARIMA models incorporate both non-seasonal and seasonal elements:

1. **Non-seasonal components**:

    - **AR (AutoRegressive)**: Uses previous values to predict future values.

- **I (Integrated)**: Involves differencing the series to achieve stationarity.
- **MA (Moving Average)**: Uses past forecast errors to make predictions.

2. **Seasonal components**:
    - **Seasonal AR (AutoRegressive)**: Considers seasonal lags of the series.
    - **Seasonal I (Integrated)**: Involves seasonal differencing of the series.
    - **Seasonal MA (Moving Average)**: Models past seasonal forecast errors.

Mathematical Formulation

A SARIMA model is denoted as SARIMA(p, d, q)(P, D, Q)[S], where p, d, q are the non-seasonal parameters, P, D, Q are the seasonal parameters, and S is the length of the seasonal cycle.

Equation of the SARIMA model:

$$(1-\sum_{i=1}^{p}\phi_i L^i)(1-\sum_{i=1}^{P}\Phi_i L^{iS})(1-L)^d(1-L^S)^D X_t = (1+\sum_{j=1}^{q}\theta_j L^j)(1+\sum_{j=1}^{Q}\Theta_j L^{jS})\epsilon_t$$

Here:

- L is the lag operator, $LX_t = X_{t-1}$.
- ϕ_i and Φ_i are the non-seasonal and seasonal AR parameters.
- θ_j and Θ_j are the non-seasonal and seasonal MA parameters.
- ϵ_t is the white noise error term.
- X_t is the time series.

Steps to Model with SARIMA

1. **Seasonal Decomposition**: Identify the seasonality in the data through plots or autocorrelation analysis.
2. **Parameter Selection**: Determine p, d, q, P, D, Q, and S based on the ACF and PACF and knowledge of the data's seasonality.
3. **Model Fitting**: Fit the model using Maximum Likelihood Estimation or conditional least squares.
4. **Diagnostic Checking**: Analyze the residuals to ensure they resemble white noise.
5. **Forecasting**: Use the model to forecast future values.

Benefits and Limitations

Benefits:

- Effective in modeling seasonal variations.
- Provides methodologies for both non-seasonal and seasonal changes.

Limitations:

- Complex model identification and parameter selection.
- Potential for overfitting with many parameters.
- Assumes stationarity after differencing.

8 Facebook Prophet: Explanation and Mathematical Foundations

Facebook Prophet is an open-source forecasting tool developed by Facebook's Core Data Science team. It provides an intuitive interface for time-series forecasting, particularly suitable for datasets with strong seasonal patterns and multiple seasonalities.

Core Components of Facebook Prophet

1. **Trend Modeling**: Prophet decomposes the time series into trend, seasonal, and holiday components. The trend component captures the overall direction of the data, allowing for both linear and non-linear trend modeling.
2. **Seasonality Modeling**: Prophet accounts for periodic patterns or seasonalities in the data, including daily, weekly, and yearly seasonality. It can handle multiple seasonalities simultaneously.
3. **Holiday Effects**: Users can specify holidays or events that might impact the time series. Prophet automatically incorporates these effects into the forecast.
4. **Uncertainty Estimation**: Prophet provides uncertainty intervals for forecasts, allowing users to quantify the uncertainty associated with each prediction.

Mathematical Formulation

Prophet uses an additive model that decomposes the time series into trend, seasonality, holiday, and error components:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

Where:

- $y(t)$ denotes the observed value of the time series at time t.
- $g(t)$ represents the trend component.
- $s(t)$ represents the seasonal component.
- $h(t)$ represents the holiday component.
- ε_t represents the error term.

The trend component $g(t)$ and seasonal component $s(t)$ are modeled using Fourier series expansions, allowing Prophet to capture complex seasonal patterns flexibly.

Benefits and Limitations

Benefits:

- **Ease of Use**: Prophet offers a simple and intuitive interface.
- **Flexibility**: Capable of handling multiple seasonalities, missing data, and outliers.
- **Uncertainty Quantification**: Provides uncertainty intervals for forecasts, aiding in decision-making.

Limitations:

- **Limited Customization**: May not support all advanced modeling techniques or custom model specifications.
- **Computationally Intensive**: Training Prophet models can be computationally demanding, especially for large datasets or complex models.