ELECTRICITY PRICE PREDICTION

PHASE 4 : Document

TEAM MEMBERS:

S. NISHA (612721104064)

S. NIVETHA (612721104067)

P. PAVITHRA (612721104069)

S. PARAMESHWARI (612721104068)

INTRODUCTION:

Predicting electricity prices in the future involves a combination of engineering, modeling, and evaluation techniques. The electricity market is influenced by a multitude of factors, including supply and demand, weather conditions, fuel prices, government policies, and more. Here are some ideas for predicting electricity price.

FUTURE ENGINEERING:

Predicting future electricity prices is a complex task that often involves various factors such a supply and demand dynamics, weather conditions, energy policies and more. Engineers and data scientists use various techniques , includes time serious analysis , machine learning , and statistical modeling , to make these predictions . These models analyze historical price data , market conditions and other relevant variables to forecast future prices. Keep in mind that accurate predictions can be challenging due to the volatility of energy and unforeseen events.

PREDICTING FUTURE ELECTRICITY PRICES THROUGH MODELLING :

1. DATA COLLECTION:

   Gather historical data on electricity prices, which should include relevant factors like demand , generation mix , weather patterns , and market trends.

2. DATA PREPROCESSING :

   Clean and preprocess the data to handle missing values , outliers and format it for modelling.

3. FEATURE ENGINEERING :

Create or select relevant features that may impact electricity prices , such as time of day . weather conditions , holidays , or economic indicators.

4. MODEL SELECTION:

Choose an appropriate modelling technique , which could include time series analysis (e.g., ARIMA or GARCH models) or machine learning algorithms (e.g., regression, neural networks or decision trees).

5. MODEL TRAINING :

Use historical data to train the chosen model. This involves learning the relationships between the features and electricity prices.

6. MODEL EVALUATION :

Assess the models' performance using metrices like Mean ABSOLUTE ERRORE (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) through cross- validation or holdout validation datasets.

7. HYPERPARAMETER TUNING:

Optimize the models' hyperparameter to improve its predictive accuracy.

8. FUTURE PRICE PREDICTION :

Apply the trained model to make predictions for future electricity precise based on new data

9. MONITORING AND UPDATES:

Continuously monitor the models performance and update it as new data becomes available or market conditions change .

Price forecasting is predicting a commodity/ product /service price by evaluating various factors like its characteristics, demand, seasonal ,trends ,other commodities prices etc.,

Analyzing Electricity Price Time Series Data using Python: Time Series Decomposition and Price Forecasting using a Vector Autoregression (VAR) Model

```
def retrieve _time _series(api, series _ID):
 """
Return the time series data frame, based on API and unique Series ID
api: API that we're connected to series _ID:
string. Name of the series that we want to pull from the EIA API
"""
 #Retrieve Data By Series ID
Series _search = api .data _by _series(series=series _ID)
 ##Create a pandas
data frame from the retrieved time series df = pd .Data Frame(series _search)
return df
 ###Execute in the main block
 #Create EIA API using your specific
  API key aPi _key = "YOR API KEY HERE"
 api = eia .API(api _key)
```

```
 #Pull the electricity price data
 series _ID='ELEC.PRICE.TX-ALL.M'
electricity _df= retrieve _time  _series(api, series _ID)
electricity _df .reset _index(level=0, in place=True)
#Rename the columns for easer analysis
electricity _df .rename(columns={'index' :'Date',
 electricity _df .columns[1]:'Electricity _Price'}, in  place=True)
```

Snapshot of the time series data for electricity prices, pulled via the EIA API

First, let's look at whether or not the monthly electricity data displays seasonality and a trend. To do this, we use the seasonal_decompose() function in the statsmodels.tsa.seasonal package. This function breaks down a time series into its core components: trend, seasonality, and random noise. The code and its outputs are displayed below:

```
def decompose_time_series(series):
  """
  Decompose a time series and plot it in the console
  Arguments:
    series: series. Time series that we want to decompose
  Outputs:
    Decomposition plot in the console
  """
  result = seasonal_decompose(series, model='additive')
  result.plot()
```

```
    pyplot.show()

#Execute in the main block

#Convert the Date column into a date object

electricity_df['Date']=pd.to_datetime(electricity_df['Date'])

#Set Date as a Pandas DatetimeIndex

electricity_df.index=pd.DatetimeIndex(electricity_df['Date'])

#Decompose the time series into parts

decompose_time_series(electricity_df['Electricity_Price'])
```

Time Series Decomposition: Monthly Electricity Prices in TX

Although we're analyzing electricity data going back to 2001 and the breakdown above is from March 2019, for the sake of simplicity we'll assume that natural gas has been one of the main sources of electricity generation in Texas for the past 15-20 years. Consequently, we're going to pull the time series of natural gas prices via the EIA API and compare it side-by-side to the TX electricity price time series:

```
#Pull in natural gas time series data

series_ID='NG.N3035TX3.M'

nat_gas_df=retrieve_time_series(api, series_ID)

nat_gas_df.reset_index(level=0, inplace=True)

#Rename the columns

nat_gas_df.rename(columns={'index':'Date',

          nat_gas_df.columns[1]:'Nat_Gas_Price_MCF'},
```

```
        inplace=True)
#Convert the Date column into a date object
nat_gas_df['Date']=pd.to_datetime(nat_gas_df['Date'])
#Set Date as a Pandas DatetimeIndex
nat_gas_df.index=pd.DatetimeIndex(nat_gas_df['Date'])
#Decompose the time series into parts
decompose_time_series(nat_gas_df['Nat_Gas_Price_MCF'])


#Merge the two time series together based on Date Index
master_df=pd.merge(electricity_df['Electricity_Price'], nat_gas_df['Nat_Gas_Price_MCF'],
                   left_index=True, right_index=True)
master_df.reset_index(level=0, inplace=True)


#Plot the two variables in the same plot
plt.plot(master_df['Date'],
         master_df['Electricity_Price'], label="Electricity_Price")
plt.plot(master_df['Date'],
         master_df['Nat_Gas_Price_MCF'], label="Nat_Gas_Price")
# Place a legend to the right of this smaller subplot.
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Natural Gas Price vs. TX Electricity Price over Time')
```

```
plt.show()
```

In the code below, each time series is transformed using the numpy natural log function, and then differenced by one interval:

```
#Transform the columns using natural log

master_df['Electricity_Price_Transformed']=np.log(master_df['Electricity_Price'])

master_df['Nat_Gas_Price_MCF_Transformed']=np.log(master_df['Nat_Gas_Price_MCF'])



#Difference the data by 1 month

n=1

master_df['Electricity_Price_Transformed_Differenced'] = master_df['Electricity_Price_Transformed'] - master_df['Electricity_Price_Transformed'].shift(n)

master_df['Nat_Gas_Price_MCF_Transformed_Differenced'] = master_df['Nat_Gas_Price_MCF_Transformed'] - master_df['Nat_Gas_Price_MCF_Transformed'].shift(n)
```

determine that a time series is stationary, the test must return a p-value of less than .05. In the Python code below, we run the Augmented Dickey-Fuller test on the transformed, differenced time series, determining that they are both stationary (the p-values returned are .000299 and 0 for the electricity and natural gas time series, respectively):

```


def augmented_dickey_fuller_statistics(time_series):

  """

  Run the augmented Dickey-Fuller test on a time series

  to determine if it's stationary.

  Arguments:
```

```
    time_series: series. Time series that we want to test

  Outputs:

    Test statistics for the Augmented Dickey Fuller test in

    the console

  """

  result = adfuller(time_series.values)

  print('ADF Statistic: %f' % result[0])

  print('p-value: %f' % result[1])

  print('Critical Values:')

  for key, value in result[4].items():

    print('\t%s: %.3f' % (key, value))

#Execute in the main block

#Run each transformed, differenced time series thru the Augmented Dickey Fuller test

print('Augmented Dickey-Fuller Test: Electricity Price Time Series')

 augmented_dickey_fuller_statistics(master_df['Electricity_Price_Transformed_Differenced'].dropna())

print('Augmented Dickey-Fuller Test: Natural Gas Price Time Series')


augmented_dickey_fuller_statistics(master_df['Nat_Gas_Price_MCF_Transformed_Differenced'].dropna()
)
```

Outputs for the Augmented Dickey-Fuller Test for the electricity price time series and the natural gas price time series, respectively

Now that we've made our two time series stationary, it's time to fit the data to a VAR model. The Python code below successfully builds the model and returns a summary of the results, where we use a 95/5 percent split for the training/validation sets:

```
#Conver the dataframe to a numpy array

master_array=np.array(master_df[['Electricity_Price_Transformed_Differenced',

                                 'Nat_Gas_Price_MCF_Transformed_Differenced']].dropna())



#Generate a training and test set for building the model: 95/5 split

training_set = master_array[:int(0.95*(len(master_array)))]

test_set = master_array[int(0.95*(len(master_array))):]



#Fit to a VAR model

model = VAR(endog=training_set)

model_fit = model.fit()

#Print a summary of the model results

model_fit.summary()
```

VAR Model Summary, with y1=Electricity Price Time Series, and y2=Natural Gas Price Time Series

What we really want to focus on in the model summary above is the equation for y1, where y1 estimates the electricity prices in the state of Texas based on lagged values of itself and lagged natural gas prices.

```
def calculate_model_accuracy_metrics(actual, predicted):
  """
  Output model accuracy metrics, comparing predicted values
  to actual values.
  Arguments:
    actual: list. Time series of actual values.
    predicted: list. Time series of predicted values
  Outputs:
    Forecast bias metrics, mean absolute error, mean squared error,
    and root mean squared error in the console
  """
  #Calculate forecast bias
  forecast_errors = [actual[i]-predicted[i] for i in range(len(actual))]
  bias = sum(forecast_errors) * 1.0/len(actual)
  print('Bias: %f' % bias)
  #Calculate mean absolute error
  mae = mean_absolute_error(actual, predicted)
  print('MAE: %f' % mae)
  #Calculate mean squared error and root mean squared error
```

```
    mse = mean_squared_error(actual, predicted)

    print('MSE: %f' % mse)

    rmse = sqrt(mse)

    print('RMSE: %f' % rmse)

#Execute in the main block

#Un-difference the data

for i in range(1,len(master_df.index)-1):

    master_df.at[i,'Electricity_Price_Transformed']= master_df.at[i-
1,'Electricity_Price_Transformed']+master_df.at[i,'Electricity_Price_Transformed_Differenced_PostProce
ss']



#Back-transform the data

master_df.loc[:,'Predicted_Electricity_Price']=np.exp(master_df['Electricity_Price_Transformed'])



#Compare the forecasted data to the real data

print(master_df[master_df['Predicted']==1][['Date','Electricity_Price', 'Predicted_Electricity_Price']])

#Evaluate the accuracy of the results

calculate_model_accuracy_metrics(list(master_df[master_df['Predicted']==1]['Electricity_Price']),

                                 list(master_df[master_df['Predicted']==1 ['Predicted_Electricity_Price']))
```

The accuracy of forecasting short- or medium-term electricity loads and prices is critical in the energy market. The amplitude and duration of abnormally high prices and load

spikes can be detrimental to retailers and production systems. Therefore, predicting these spikes to effectively manage risk is critical

.

An electricity price multi-bi-forecasting method is proposed using a new MIMO scheme that exports both the PF and IF results. A multivariable forecasting scheme is adopted using electricity price data combined with electric load data.

Load-forecasting methods can be classified into two broad categories: parametric methods and artificial intelligence–based methods. The artificial intelligence methods are further classified into neural network–based methods and fuzzy logic–based methods.

EVALUTION OF ELECTRICITY PRICE PREDICTION:

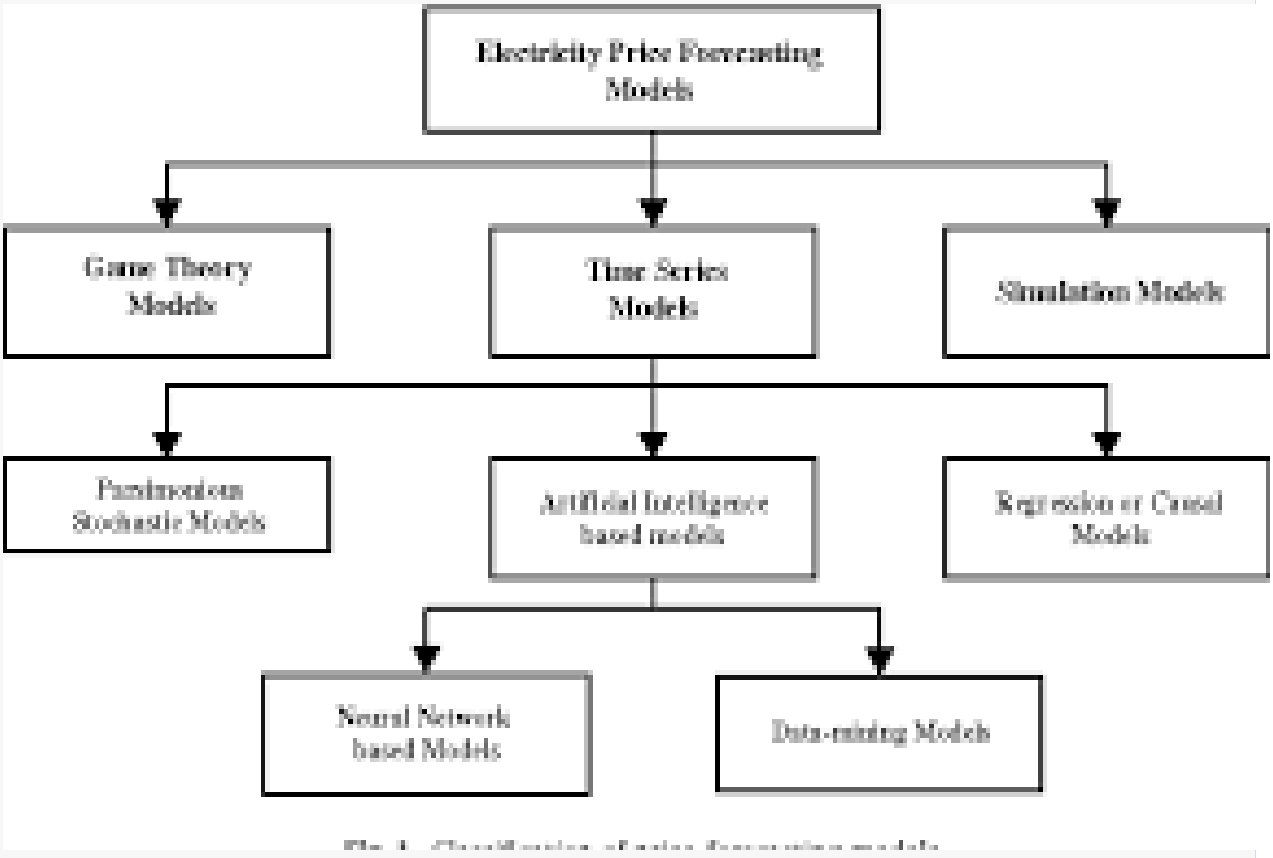
The evaluation of electricity price prediction models is crucial to assess their accuracy and usefulness. There are several commonly used metrics and methods to evaluate the performance of these models:

1. **Mean Absolute Error (MAE):** MAE measures the average absolute difference between predicted and actual prices. It gives an idea of the magnitude of errors.
2. **Mean Squared Error (MSE):** MSE measures the average of the squared differences between predicted and actual prices. It amplifies the impact of larger errors.
3. **Root Mean Squared Error (RMSE):** RMSE is the square root of MSE. It provides a measure of the average error in the same unit as the target variable (electricity price).
4. **Mean Absolute Percentage Error (MAPE):** MAPE calculates the average percentage difference between predicted and actual prices. It's useful for understanding the relative error.
5. **R-squared (R^2):** R-squared measures the proportion of the variance in the dependent variable (electricity price) that is predictable from the independent variables (features) in the model. A higher R-squared indicates a better fit. **Adjusted R-squared:** Adjusted R-squared is a modified version of R-squared that accounts for the number of predictors in the model. It helps prevent overfitting by penalizing the inclusion of unnecessary variables.
6. **Residual Analysis:** Examining the distribution of residuals (the differences between predicted and actual values) can reveal patterns or biases in the model's predictions. A good model

should have residuals that are approximately normally distributed, centered around zero, and exhibit homoscedasticity (constant variance).

7. **Time Series Analysis:** For electricity price prediction, you may need to use time series-specific metrics, such as AIC (Akaike Information Criterion) or BIC (Bayesian Information Criterion), which help evaluate the quality of time series models.
8. **Cross-Validation:** Use techniques like k-fold cross-validation to assess how well the model generalizes to new data. This helps prevent overfitting and provides a more realistic evaluation of the model's performance.
9. **Out-of-Sample Testing:** Evaluate the model's performance on a separate test dataset that it has not seen during training. This simulates how the model will perform in real-world scenarios.
10. **Domain Expert Evaluation:** Consult domain experts who have knowledge of the electricity market to get qualitative feedback on the model's performance and its practical utility.
11. **Benchmarking:** Compare the model's performance with existing industry benchmarks or with simpler forecasting methods to gauge its improvement.
12. **Economic Impact Assessment:** In some cases, it might be useful to assess the economic impact of using the model for electricity price prediction, such as potential cost savings or revenue gains.

Keep in mind that the choice of evaluation metrics and methods can vary depending on the specific characteristics of your dataset, the type of model you're using (e.g., regression, time series, machine learning), and the goals of your electricity price prediction task. It's essential to consider the context and requirements of your application when selecting the mostappropriate evaluation approach.



Typical data sources for electricity price predictions include a combination of historical energy market data, environmental data, and other relevant information. Here are some common data sources:

1. **Historical Electricity Price Data:**
    - Hourly, daily, or sub-hourly electricity price data from energy markets, which is often publicly available. This data is essential for training predictive models and understanding price trends.
2. **Load Data:**
    - Historical electricity load data, which represents the amount of electricity consumed over time. Load data is critical for understanding demand patterns and their impact on prices.
3. **Generation Data:**
    - Information on power generation sources, including data on the availability and output of various energy sources such as coal, natural gas, nuclear, renewables (e.g., wind and
4. **Weather Data:**
    - Meteorological data, including temperature, humidity, wind speed, and solar irradiance. Weather conditions can significantly affect electricity supply and demand.
5. **Market Data:**
    - Data related to market fundamentals, such as supply and demand forecasts, infrastructure changes, grid conditions, regulatory changes, and market rules.
6. **Fuel Price Data:**
    - Data on the prices of fuels used in electricity generation, such as coal, natural gas, and oil. These

prices can impact the cost of generation and, subsequently, electricity prices.

7. **Transmission and Grid Data:**
   - Information about the state of the electricity grid, transmission constraints, and grid stability. This data can influence electricity prices due to transmission losses and congestion.
8. **Economic Indicators:**
   - Economic data, such as GDP growth, employment rates, and industrial activity, which can provide insights into overall energy consumption and its relationship to economic conditions.

9. **Publicly Available Reports:**
   - Reports and publications from energy market operators, regulatory bodies, and industry associations can provide valuable insights into market conditions and forecasts.
10. **Energy Exchange Data:**
    - Data from energy exchanges and trading platforms that provide information on real-time or future energy prices, trading volumes, and market liquidity.
11. **Financial Data:**
    - Data related to financial markets, including futures and options prices, can provide additional insights into market expectations.

12. **Machine Learning Features:**

- Engineered features specific to machine learning models, such as lagged variables, moving averages, and technical indicators, to capture patterns in the data.
- Predictions and data related to renewable energy availability, especially for wind and solar generation. These forecasts are crucial for understanding the variable nature of renewables.

13. **Regulatory and Policy Information:**

- Information on government policies, subsidies, tax incentives, and regulations affecting the energy sector. Changes in policy can have a substantial impact on electricity prices.

14. **Market Sentiment and News Data:**

Social media sentiment analysis, news articles, and public sentiment data can help capture public perception, market sentiment, and speculative factors that influence energy prices.

Data sources may vary based on the specific goals of your electricity price prediction project, the region and market you are focused on, and the availability of data. Data from multiple sources is often combined and preprocessed to build predictive models

-
-