Institute of Technology of Cambodia

Department of Applied Mathematics and Statistics

PROJECT OF INTRODUCTION TO DATA SCIENCE
GROUP: I3-AMS-B (6)

Lecturers: Dr. PHAUK Sokkhey (Course)
Mr. PEN Chentra (TP)

Name of Students	ID of Students	Score
MON SREYLIN	e20221701	…….
PHYRUN PICHCHHORDA	e20220895	…….
SOM CHANN REAKSMEY	e20220981	…….
SONGSEANG PISEY	e20220225	…….
SOPHON RACHANA	e20220725	…….

Academic Year 2024-2025

Table of Content

I. Introduction

a. Project Overview

The prediction of electricity prices plays a vital role in managing the energy sector efficiently. This project focuses on building models to analyze and predict electricity prices in the United States using historical data. By identifying key factors such as price, revenue, and sales across various sectors and states, the study aims to provide accurate and interpretable predictive models that can guide multiple stakeholders.

i. Importance of Predicting Electricity Prices

Electricity prices are influenced by numerous factors, including demand, resource availability, policies, and economic trends. Predicting these prices can:

- **Help Businesses**: Companies can optimize energy consumption, manage budgets, and plan future operations more effectively.
- **Assist Policymakers**: Insights from price trends allow for better energy regulation and the development of policies to ensure energy affordability.
- **Support Investors**: Accurate predictions enable strategic investment decisions in the energy market, promoting sustainable and profitable ventures.

ii. Objectives of the Study

- **Trend Analysis**: Understand historical changes in electricity prices and related metrics.
- **Sectoral Insights**: Compare the behavior of electricity prices across sectors like residential, commercial, industrial, and transportation.
- **Price Forecasting**: Leverage machine learning techniques to predict future electricity prices and assist in planning.

By leveraging statistical and machine learning techniques, this project seeks to offer actionable insights that can be utilized for efficient energy management and decision-making.

b. Data Description

The dataset provided is from the kaggle site that contains various information about many sectors across different states in the United States. The data spans multiple years and months, capturing key metrics such as price, revenue, and sales for each sector. The columns in the dataset are:

- Year: The year recorded from 2001 to 2024
- Month: The month recorded from January to December
- StateDescription: The name of the state in the US
- SectorName: The name of the sector (commercial, industrial, residential, transportation).
- Price: The price of the electricity prices according to sectorName stateDescription and time period.
- revenue: The revenue according to sectorName, stateDescription and time period.
- sales: The sales figures for the sector in the given state and time period.

i. Dataset Features and Relevance

- The dataset includes 85,870 entries, providing a robust basis for analysis and modeling.
- **Numeric Columns**: Variables such as price, revenue, and sales provide quantitative insights.
- **Categorical Columns**: Variables like state and sector help in segmenting the data for deeper analysis.

ii. Applications of the Dataset

The dataset's structure and breadth make it suitable for:

- **Trend Analysis**: Observing historical changes in electricity prices, revenue, and sales.

- **Sectoral Comparisons**: Understanding differences in pricing and performance across sectors.
- **Predictive Modeling**: Building models to forecast electricity prices and plan for the future.

II. Exploratory Data Analysis (EDA)

a. Data Cleaning

This dataset contains 85870 entries from range 0 to 85869 with 8 columns.
And there is a total sum of 26040 null values in the customers column.

```
dt.isnull().sum()
```
✓ 0.0s Python

```
year                   0
month                  0
stateDescription       0
sectorName             0
customers          26040
price                  0
revenue                0
sales                  0
dtype: int64
```

Therefore, the customer's column is dropped.

```
dt = dt.drop(columns=['customers'])
```
Python

```
dt.columns
```
Python

```
Index(['year', 'month', 'stateDescription', 'sectorName', 'price', 'revenue',
       'sales'],
      dtype='object')
```

```
dt.describe()
```

	year	month	price	revenue	sales
count	85870.000000	85870.000000	85870.000000	85870.000000	85870.000000
mean	2012.043321	6.480144	9.300193	586.627155	5980.048970
std	6.660304	3.461589	5.010382	2161.047702	21302.453181
min	2001.000000	1.000000	0.000000	-0.000010	0.000000
25%	2006.000000	3.000000	6.650000	29.475195	289.144572
50%	2012.000000	6.000000	8.840000	121.641500	1447.518085
75%	2018.000000	9.000000	11.380000	421.320628	4339.950965
max	2024.000000	12.000000	116.670000	52361.450970	391900.008970

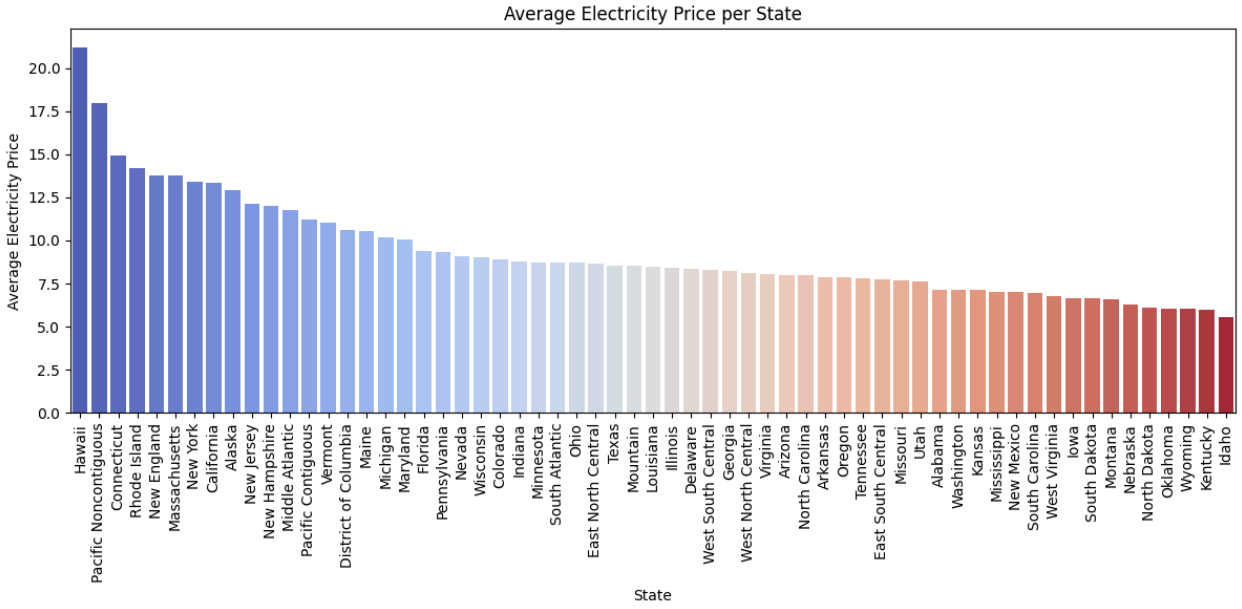
The dataset contains only 2 negative revenue values, which account for an insignificant percentage of the total data. These values can be retained for completeness without significantly impacting overall trends.

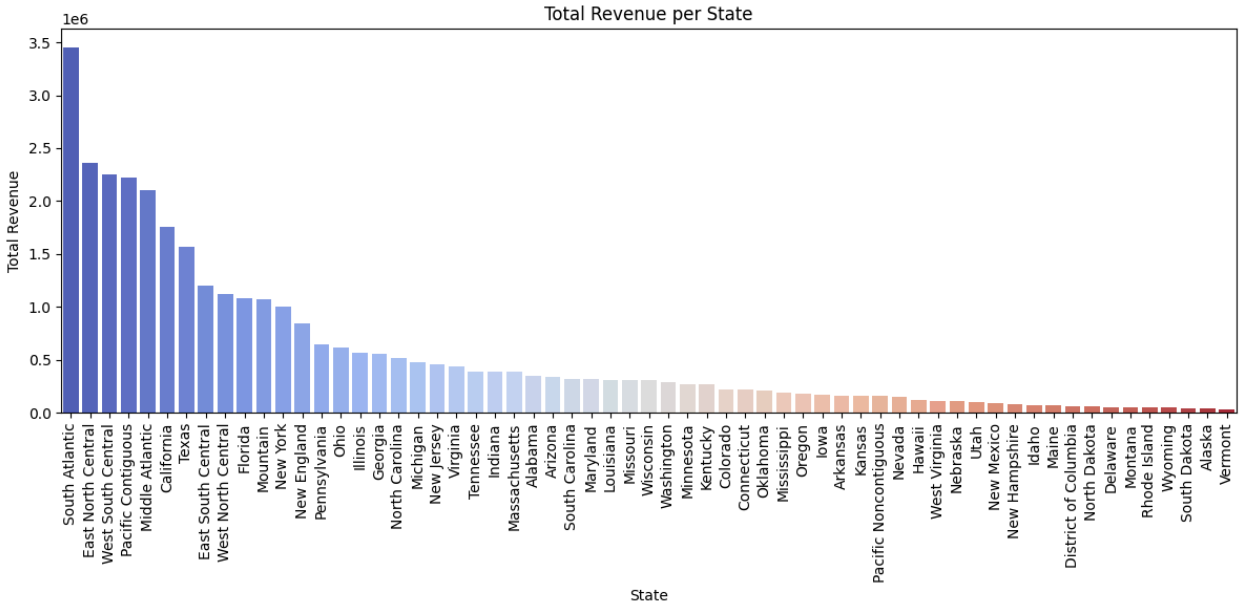
```
[48] # Count negative revenue values
     negative_count = (dt['revenue'] < 0).sum()

     print(f"Number of negative revenue values: {negative_count}")
```

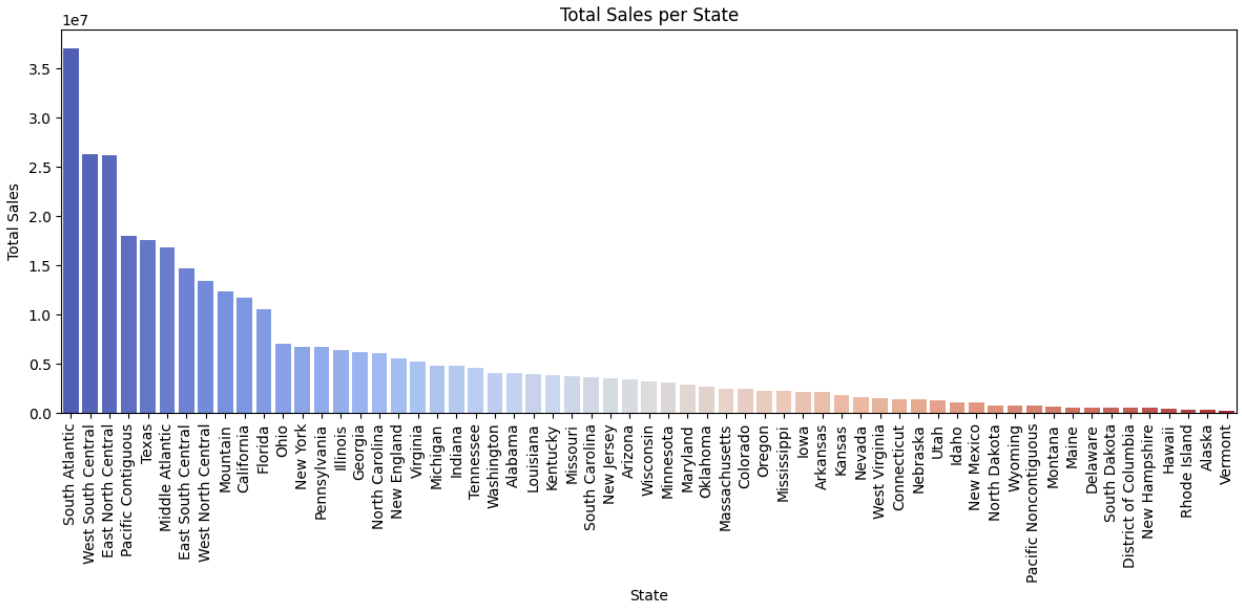
```
Number of negative revenue values: 2
```

b. Visualizing Data

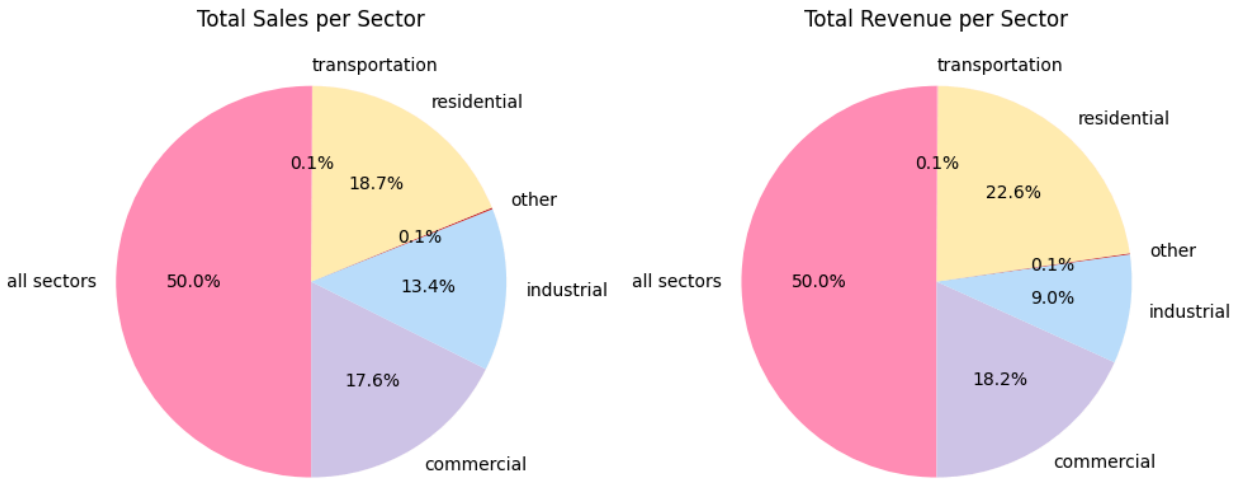
i. Visualize the relationship between the categorical variable (states) and the numerical variables (average price, total revenue, and total sales)







ii. Visualize categorical variables (sectors) and their corresponding numerical values (sales and revenue)



III. Data Preparation

a. Choosing importance features

Before building the prediction model, it's necessary to choose the relevant features for the model. By doing this, the model will be robust and efficient to leverage the strengths of ensemble learning. So, we choose Random Forest Regressor and RFE to test for the effective features.

i. Random Forest Regressor

We decided to use Random Forest Regressor because it is:

- Handling Non-Linearity:
    - Random forests can capture complex, non-linear relationships between features and the target variable, which is often the case in real-world data.

- Robustness to Overfitting:
    - By averaging the results of multiple decision trees, random forests reduce the risk of overfitting compared to individual decision trees.

- Feature Importance:
    - Random forests provide insights into feature importance, helping to identify which features are most influential in making predictions. This can guide further feature selection and engineering.

- Handling Missing Values
    - Random forests can handle missing values internally, making them more robust to incomplete data.

- Scalability:
    - Random forests can be parallelized, making them scalable to large datasets.

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()
model.fit(dt[['price', 'revenue', 'sales']], dt['price'])
print(model.feature_importances_)
```

✓ 12.0s Python

```
[9.98990346e-01 7.34540402e-04 2.75113378e-04]
```

From the code above, we determine 3 features to use for building the predictive model for price. There are prices, revenue and sales. By using feature_importances_, we see that the best feature is price because it has the biggest number compared to revenue and sales. The second relevant feature is revenue and sales is the least important one.

ii. RFE and Linear Regressor

After using Random Forest Regressor, we also use Linear Regression and RFE to define the best features for our model.
The importances of the RFE are:

- Feature Selection:
  - RFE helps in selecting the most important features by recursively considering smaller sets of features. This can improve model performance and reduce overfitting.

- Model Interpretability:
  - By reducing the number of features, RFE makes the model simpler and easier to interpret.

- Improved Performance:
  - Removing irrelevant or less important features can lead to better model performance.

```python
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

estimator = LinearRegression()
selector = RFE(estimator, n_features_to_select=3)  # Select top 5 features
selector = selector.fit(dt[['price', 'revenue', 'sales']], dt['price'])
print(selector.support_)  # True for selected features, False otherwise
```

✓ 0.0s Python

```
[ True  True  True]
```

From the above code, we use Linear Regression with RFE to define the relevant features. And we set the selected features to 3. The output is all Trues which mean all the features(e.g. price, revenue and sales) are selected or relevant.

b. Splitting Data into Training and Testing Sets

- Define feature X and variable target Y:
  - X contains the features that'll use for prediction
  - Y the target variable (price).

```python
# Define features (X) and target variable (y)
X = dt[['price', 'revenue', 'sales']]  #features
y = dt['price']  #target variable
```

- Split Data : The data is split into training and testing sets using 80-20 split.
- Splitting ensures that the model generalizes well to unseen data, reducing the risk of overfitting.

```python
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

IV. Model Building

a. Linear Model

Strengths:

- Simplicity: Easy to implement and interpret.
- Efficiency: Quick training and prediction, suitable for smaller datasets.
- Baseline Comparison: Serves as a benchmark for evaluating more complex models.

Limitations:

- Linear Assumption: Assumes a linear relationship between features and the target variable, which may not always hold.
- Sensitivity to Outliers: Linear Regression is affected by extreme values, which can distort predictions.

After splitting data into X and y for training and testing, we choose the Linear Regression model for training.

```
# Initialize and train a model
model = LinearRegression()
model.fit(X_train, y_train)
```

Then, using the trained model to predict electricity prices for the testing dataset.

```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

b. Evaluate Model

```
# Evaluate the model (example metrics)
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}")
```

Output of the mean squared error and R-squared are:

```
Mean Squared Error: 8.171903547157617e-31
R-squared: 1.0
```

The result of the mean squared error and the R-squared above indicated that our predictive model is close to the actual model and well-fitted.

Analyze Error Terms Analysis

The error distribution is centered around 0 indicates that the model predictions are very close to the actual values. The Error scale is exceptionally small and the density is concentrated tightly around 0.

Plot Predicted Price and Actual Price

The points lie closely along the diagonal line, indicating that the predicted values are very close to the true values. This suggests that the model's predictions are accurate. And in this plot, it seems there is a potential outlier at the upper end.

c. Train and Predict for each Features

First, create a function for training and predict for each feature. Filters the dataset to focus on the selected state (e.g., Alabama). Segments the filtered data by unique sectorName to build sector-specific models.

```
def train_and_predict_for_each_feature(data, state, target_feature, model_type="linear"):
    """Trains and predicts a target feature for each sector in a given state using various models.
    """
    state_data = data[data['stateDescription'] == state]
    sectors = state_data['sectorName'].unique()
```

Then, determine the future date that we want to predict and make sure that it's in datetime type. And create an empty array of predictions.

```
future_dates = pd.date_range(start='2025-01-01', end='2025-12-01', freq='MS')
predictions = []
```

For each sectorName, we define X features and y for target variables.

```
# Define features and target variable
X = sector_data[['year', 'month']]
y = sector_data[target_feature]
```

Then, allow flexibility in choosing the most suitable model for the dataset characteristics. There are 3 supported models: Linear Regression, Random Forest and XGboost.

```
if model_type == "linear":
    model = LinearRegression()
elif model_type == "randomforest":
    model = RandomForestRegressor()
elif model_type == "xgboost":
    model = XGBRegressor()
else:
    raise ValueError("Invalid model_type. Choose from 'linear', 'randomforest', or 'xgboost'.")
```

Lastly, We fit our model, make predictions and append each predictive data into the predictions array.

```
        model.fit(X, y)

        future_X = pd.DataFrame({'year': [2025] * 12, 'month': range(1, 13)})
        predicted_values = model.predict(future_X)

        for i in range(len(future_dates)):
            predictions.append([state, sector, future_dates[i].strftime('%Y-%m'), predicted_values[i]])

    return pd.DataFrame(predictions, columns=['stateDescription', 'sectorName', 'date', f'predicted_{target_feature}'])
```

Linear Regression:

- Strengths: Simplicity and interpretability.
- Limitations: May struggle with non-linear relationships between features.

Random Forest Regressor:

- Strengths: Captures non-linear patterns and provides robust predictions.
- Limitations: Computationally intensive for large datasets.

XGBoost:

- Strengths: High accuracy, handles missing data, and scales well for large datasets.
- Limitations: Requires hyperparameter tuning for optimal performance.

V. Result

After creating functions for the predictive model, we use the function for predicting price, revenue and sales for each state in the US and group by each unique sectorName.

```python
states = dt['stateDescription'].unique()
all_predictions = []

for state in states:
    price_predictions = train_and_predict_for_each_feature(dt, state, "price")
    revenue_predictions = train_and_predict_for_each_feature(dt, state, "revenue")
    sales_predictions = train_and_predict_for_each_feature(dt, state, "sales")

    all_predictions.append(price_predictions)
    all_predictions.append(revenue_predictions)
    all_predictions.append(sales_predictions)

final_predictions = pd.concat(all_predictions)
print(final_predictions)
final_predictions.to_csv('final_predictions.csv', index=False)
```

Python

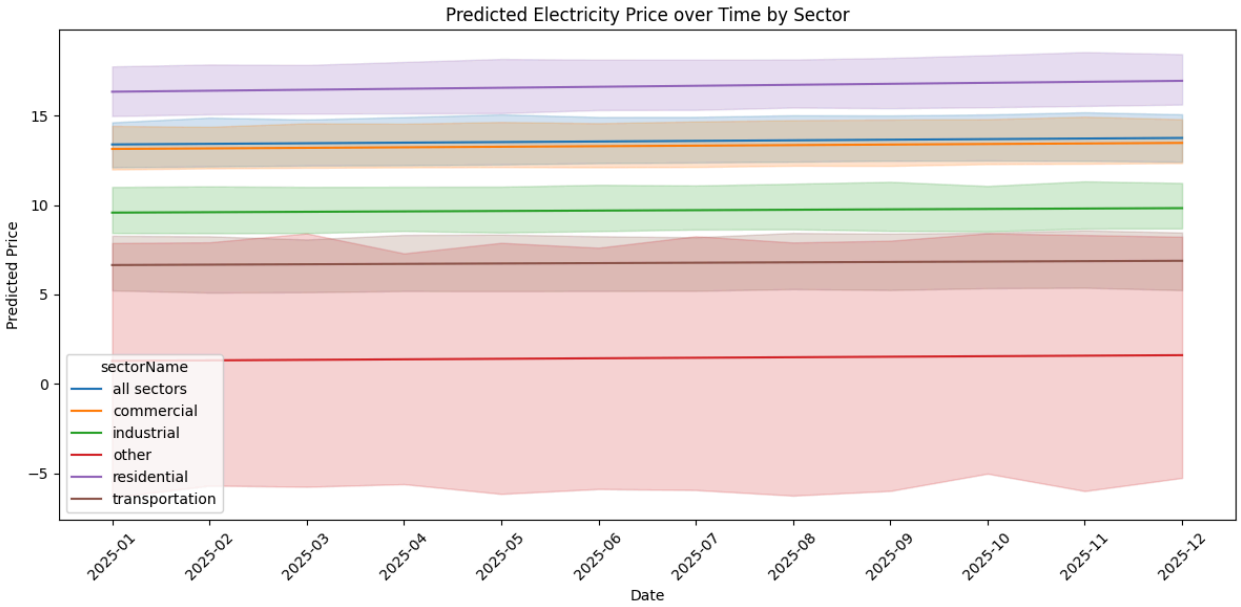
Then, create combined_predictions.csv for better visual and future predictions.

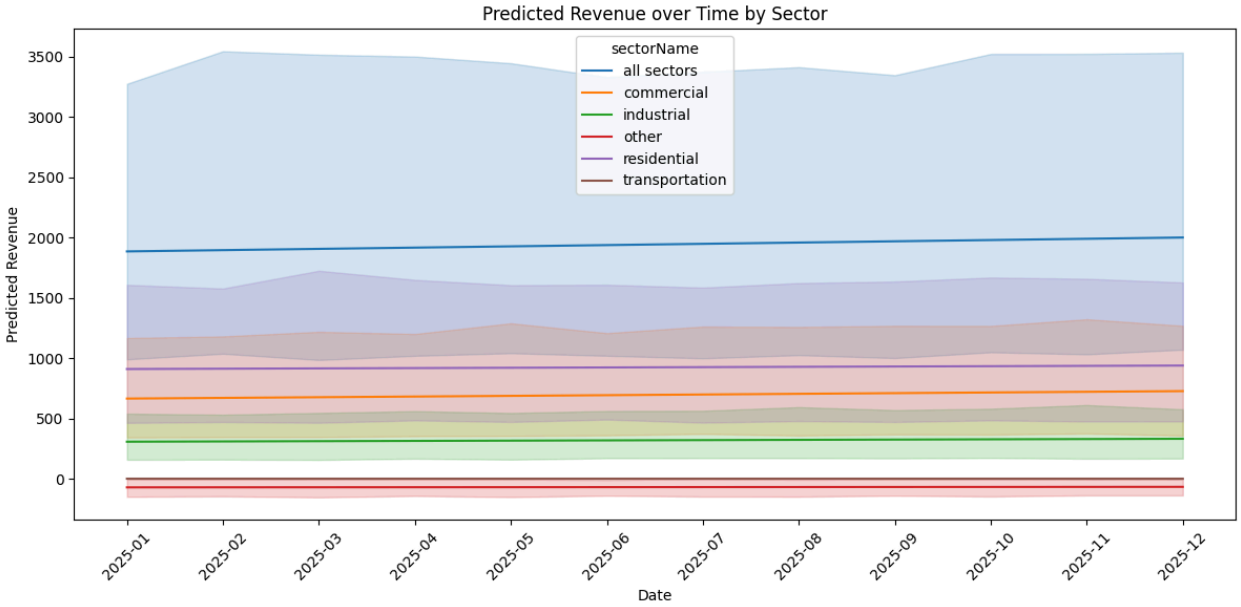
```
def combine_predictions(df):
    """Combines predictions for price, revenue, and sales into a single row per date and sector."""

    combined_data = []

    for (state, sector, date), group in df.groupby(['stateDescription', 'sectorName', 'date']):
        price = group[group['predicted_price'].notnull()]['predicted_price'].iloc[0]  # Handle potential NaNs
        revenue = group[group['predicted_revenue'].notnull()]['predicted_revenue'].iloc[0]
        sales = group[group['predicted_sales'].notnull()]['predicted_sales'].iloc[0]

        combined_data.append([state, sector, date, price, revenue, sales])

    return pd.DataFrame(combined_data, columns=['stateDescription', 'sectorName', 'date', 'predicted_price', 'predicted_revenue

combined_df = combine_predictions(final_predictions)
print(combined_df)

combined_df.to_csv('combined_predictions.csv', index=False)
```

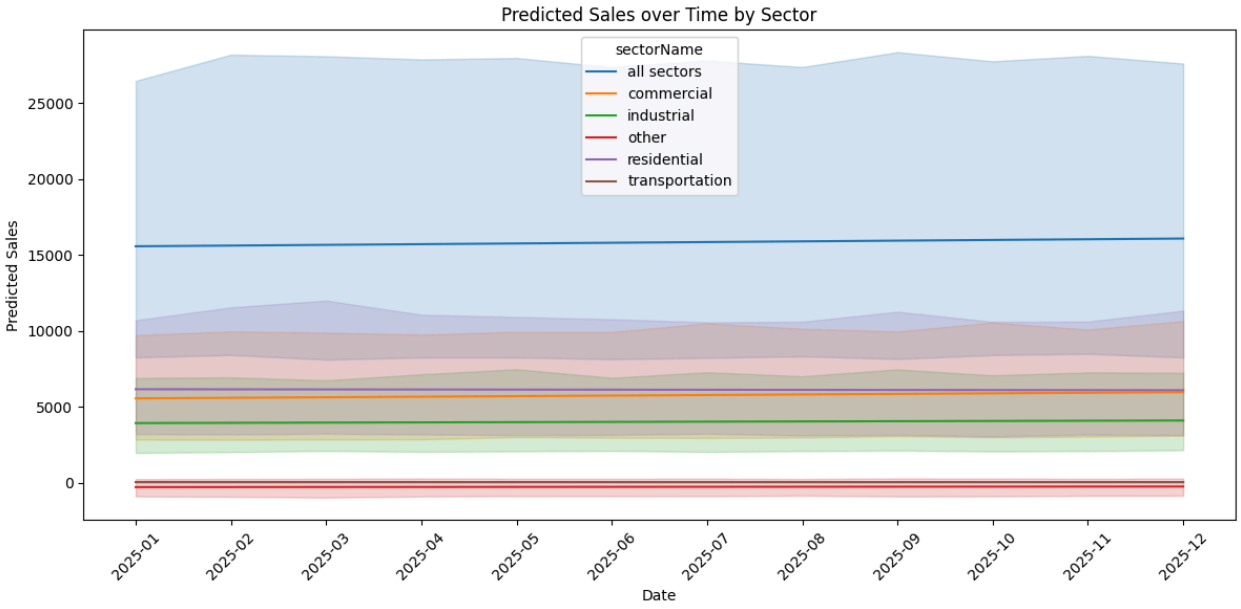
Prediction for each state in 2025

	stateDescription	sectorName	date	predicted_price	predicted_revenue	predicted_sales
0	Alabama	all sectors	2025-01	11.618125	843.626369	7273.747016
1	Alabama	all sectors	2025-02	11.650562	847.329642	7288.297147
2	Alabama	all sectors	2025-03	11.682998	851.032914	7302.847279
3	Alabama	all sectors	2025-04	11.715434	854.736186	7317.397410
4	Alabama	all sectors	2025-05	11.747871	858.439459	7331.947542
...	...	...	...	...	...	...
4459	Wyoming	transportation	2025-08	0.000000	0.000000	0.000000
4460	Wyoming	transportation	2025-09	0.000000	0.000000	0.000000
4461	Wyoming	transportation	2025-10	0.000000	0.000000	0.000000
4462	Wyoming	transportation	2025-11	0.000000	0.000000	0.000000
4463	Wyoming	transportation	2025-12	0.000000	0.000000	0.000000

4464 rows × 6 columns


Predicted Electricity Price over Time by Sector
Predicted Price
Date
sectorName
all sectors
commercial
industrial
other
residential
transportation
15
10
5
0
−5
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
2025-12


Predicted Revenue over Time by Sector
Predicted Revenue
Date
sectorName
all sectors
commercial
industrial
other
residential
transportation
3500
3000
2500
2000
1500
1000
500
0
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
2025-12



VI. Discussion and Conclusion

- ➢ Sector-Specific Trends:
  - Residential sectors generally exhibit higher prices and revenues due to consistent demand patterns.
  - Commercial and industrial sectors show lower prices, reflecting bulk energy usage discounts.
- ➢ State-Level Insights:
  - The model's predictions align with historical data trends, capturing seasonal and sectoral variations accurately.
- ➢ Seasonality:
  - Predictions highlight potential seasonal peaks, especially in the summer and winter months.

This project has successfully demonstrated the use of machine learning techniques in predicting electricity prices across different sectors and states in the United States. By leveraging historical data on price, revenue, and sales, we have developed models that not only predict future prices but also provide sector-specific insights and state-level trends. These predictions are vital for stakeholders, including businesses, policymakers, and investors, to make informed decisions.

Our exploration began with a thorough data cleaning and exploratory analysis, ensuring the robustness of the dataset. We then applied Random Forest and Recursive Feature Elimination (RFE) with a Linear Regressor to select the most impactful features. These steps allowed us to build predictive models that are both accurate and interpretable.

The models' predictions, particularly in predicting seasonal peaks in electricity prices, showcase their practical utility in real-world scenarios. For instance, the higher prices in residential sectors during summer and winter months highlight the influence of seasonal demand fluctuations. Conversely, the lower prices observed in commercial and industrial sectors emphasize the benefits of bulk energy usage.

In conclusion, the project not only supports the importance of predictive analytics in the energy sector but also opens avenues for further research. Future work could explore more complex models, incorporate additional predictors, and possibly extend the analysis to include real-time data integration for dynamic forecasting. This approach could further enhance the accuracy and relevance of the predictive models in this rapidly evolving market.

References

Dataset: https://www.kaggle.com/datasets/alistairking/electricity-prices

Linear Regression : https://www.geeksforgeeks.org/simple-linear-regression-in-python/