CS 6065 - Intro to Cloud Computing

Final Group Project

Energy Consumption Prediction and Optimization for Smart Homes

By

Samraysh Pellakur – M16474470

Sri Mani Sravika Nuthalapati - M16162004

Tulasi Rama Raju Chittiraju – M16407796

Link to Web App: https://finalproject-dataentry.azurewebsites.net/

1) Write-Up on ML Models

i. Linear Regression

Linear regression is a statistical method used for predicting a continuous dependent variable based on one or more independent variables. It assumes a linear relationship between the dependent and independent variables. The model estimates the coefficients of the linear equation by minimizing the difference between predicted and actual values. It's simple, interpretable, and often used for forecasting and trend analysis.

ii. Random Forest

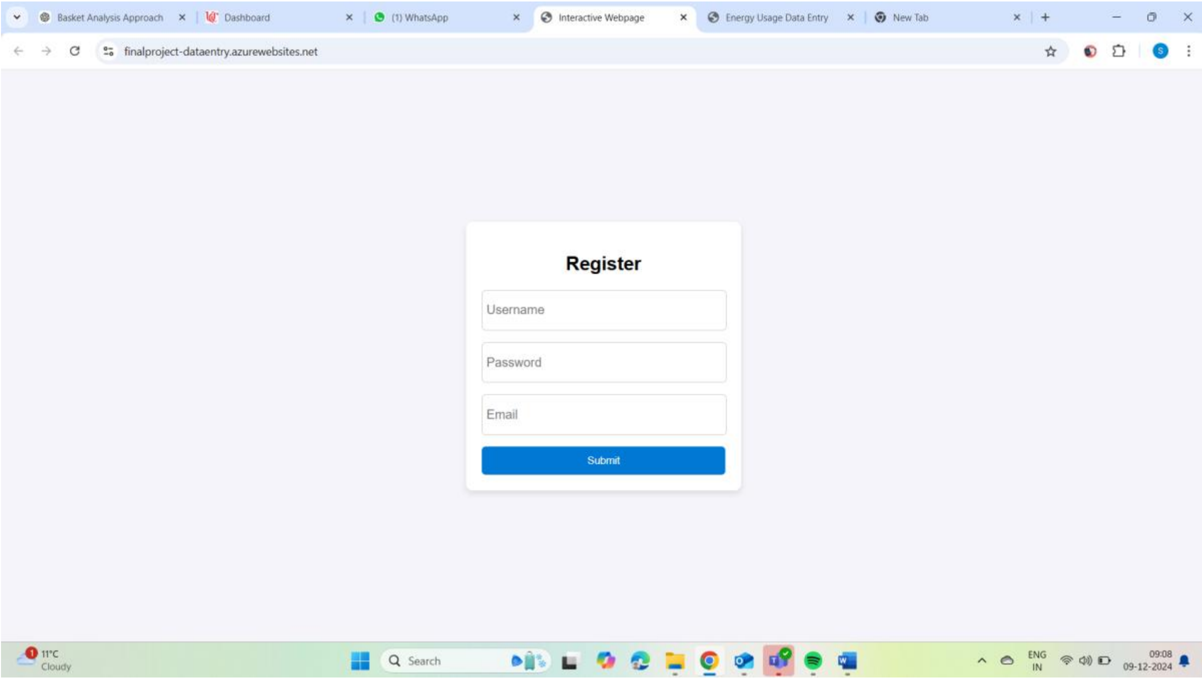
Random Forest is an ensemble learning technique that builds multiple decision trees during training and combines their predictions to improve accuracy and prevent overfitting. It works by randomly selecting subsets of features and data points, creating diversity in the trees. Each tree independently makes predictions, and the final output is determined by aggregating results. It is effective for handling large datasets and capturing complex relationships.

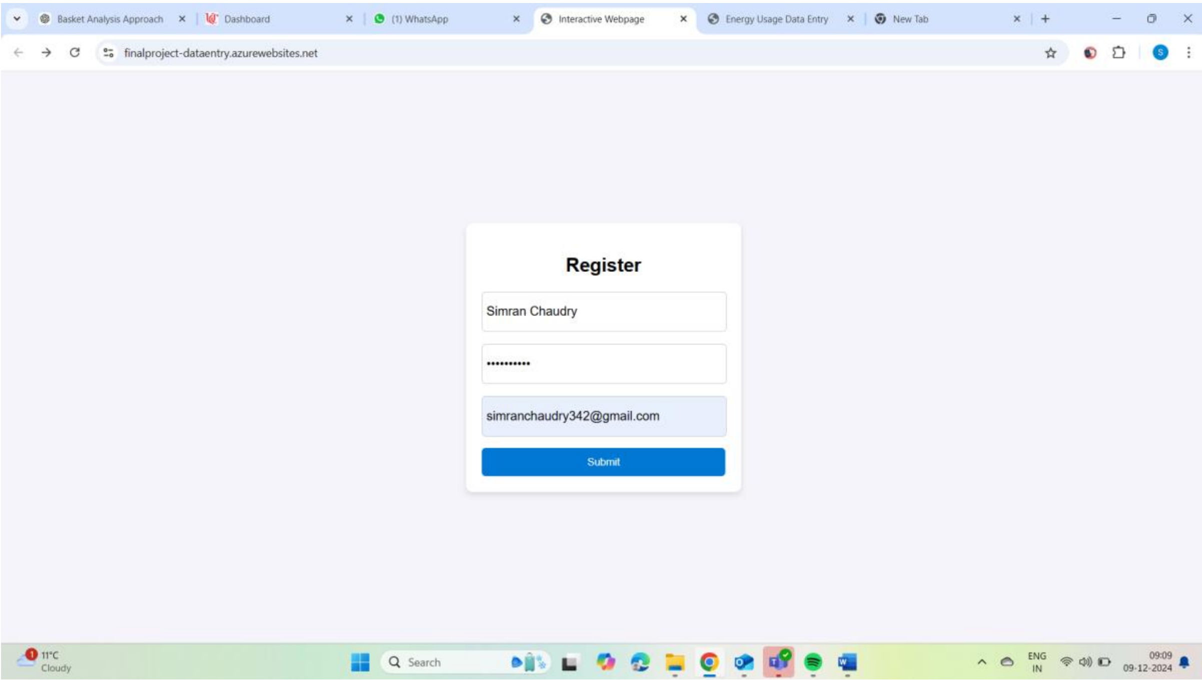
iii. Gradient Boosting

Gradient Boosting is another ensemble method that builds decision trees sequentially, where each tree attempts to correct the errors of the previous one. It focuses on minimizing the residual errors by adding weak learners one at a time. It is a powerful technique for predictive modeling and is known for its high predictive accuracy, but it can be sensitive to noise and prone to overfitting without proper tuning.

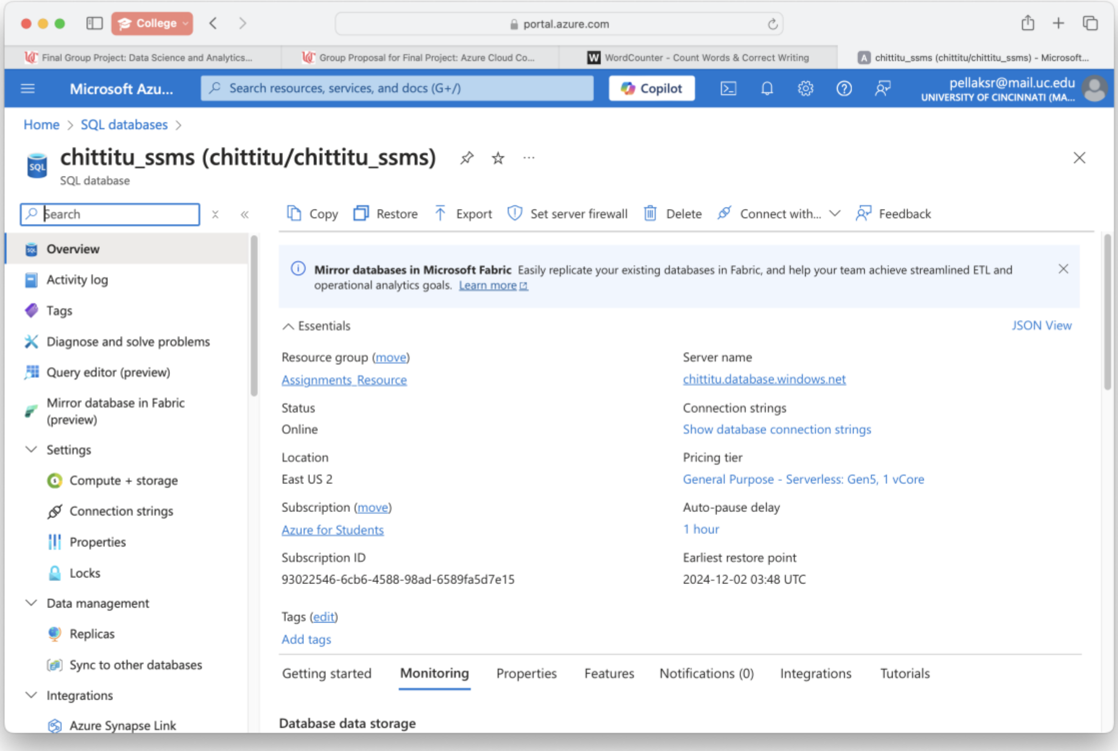
The modelling question is "Predict the energy consumption (energy_consumption_kWh) of a home based on factors such as temperature setting (temperature_setting_C), occupancy status (occupancy_status), appliance used (appliance), usage duration (usage_duration_minutes), and day of the week (day_of_week)." and we will use random forest to prepare the answer for the above question.

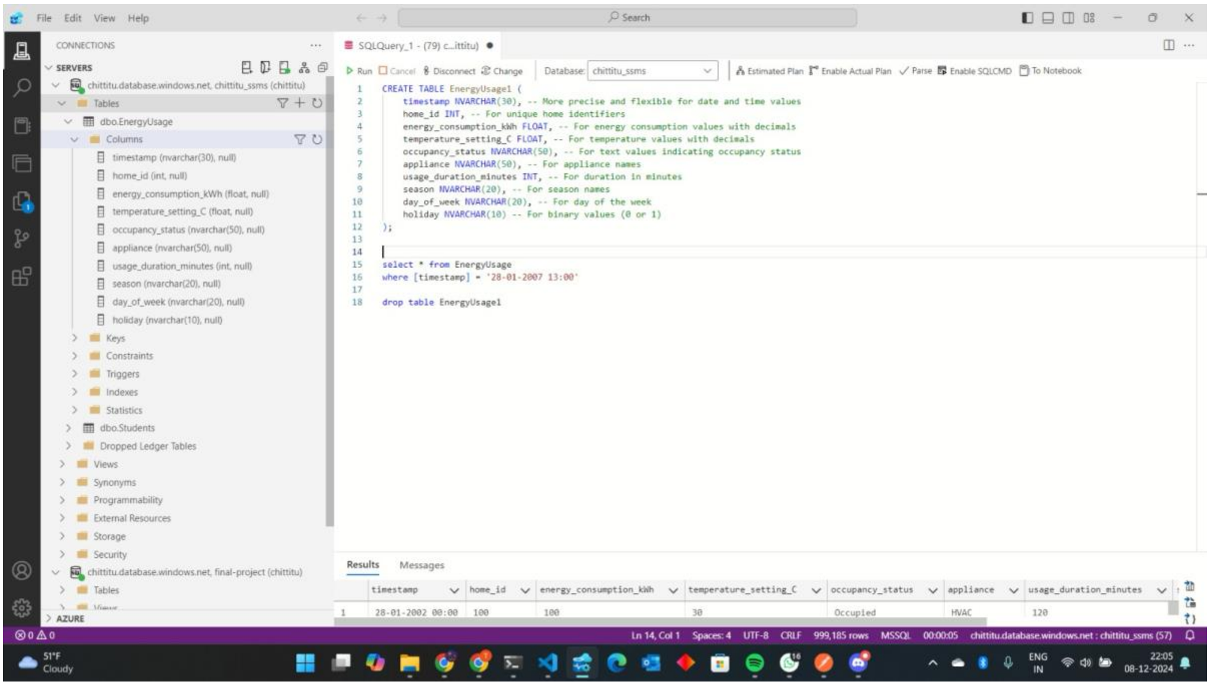
2) Web Server Setup



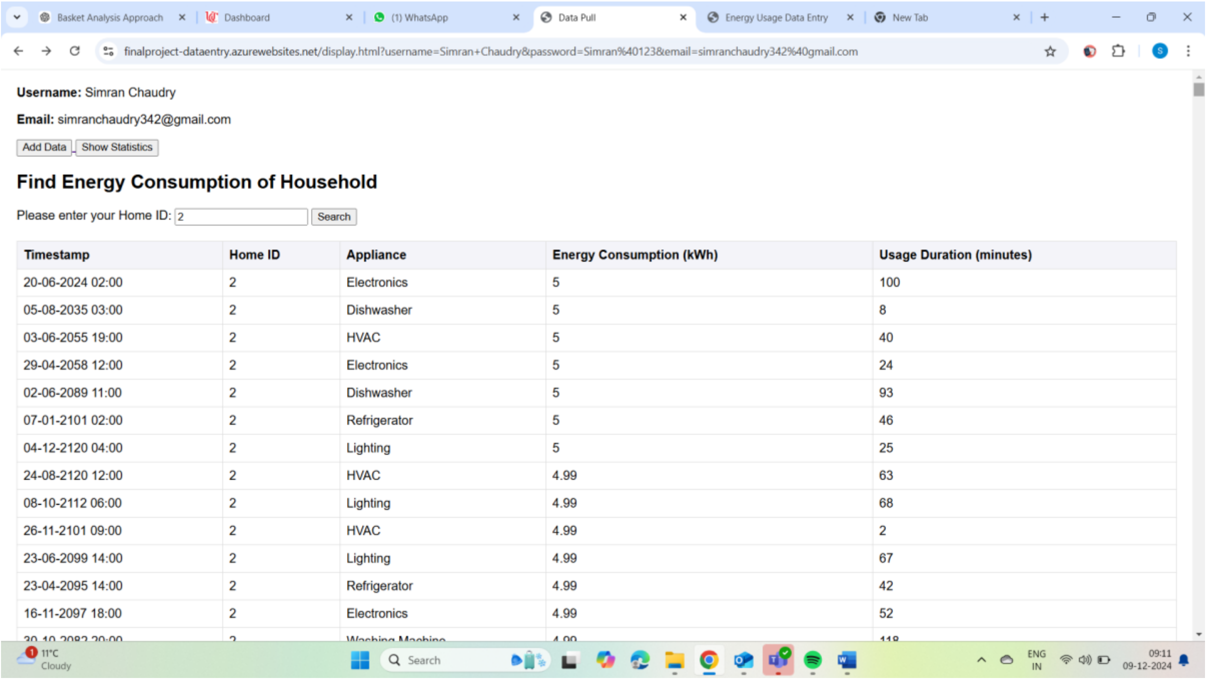


3) Datastore and Data Loading



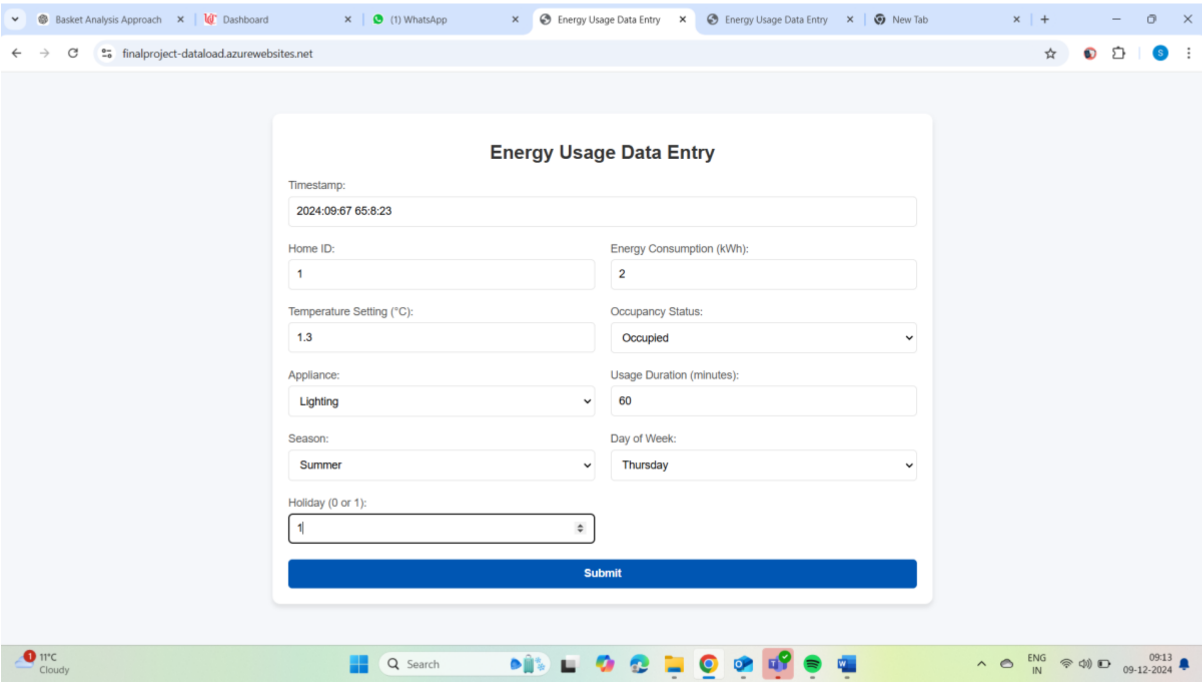


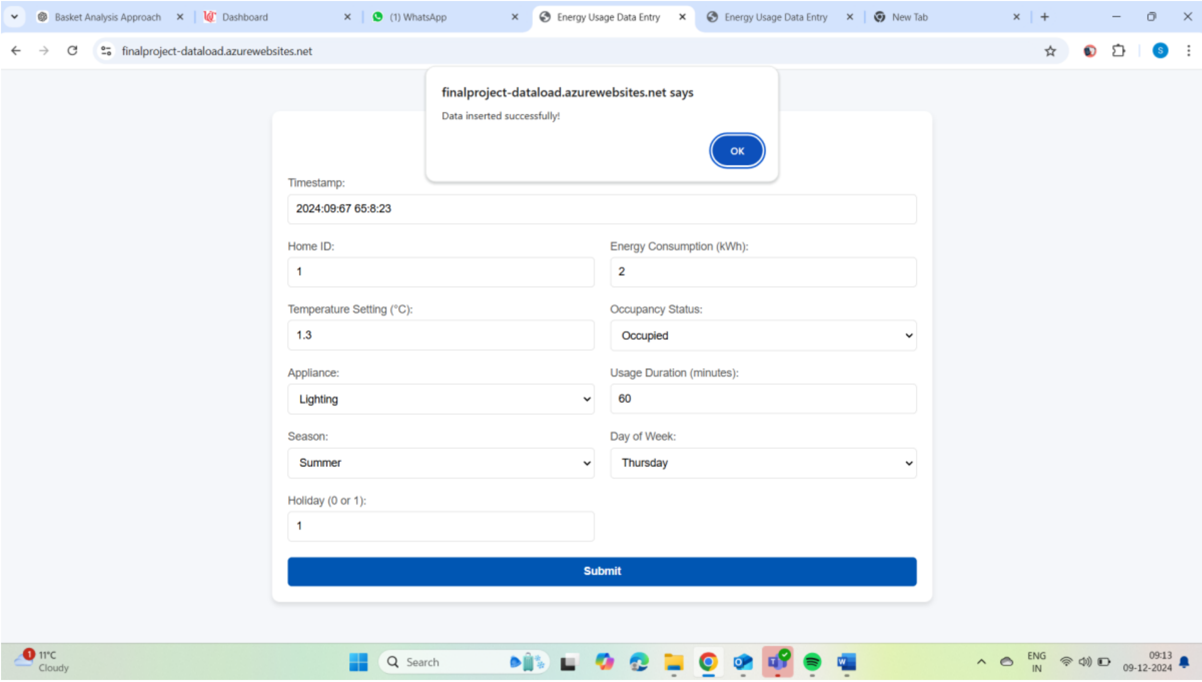
4) Interactive Web Page



Timestamp	Home ID	Appliance	Energy Consumption (kWh)	Usage Duration (minutes)
20-06-2024 02:00	2	Electronics	5	100
05-08-2035 03:00	2	Dishwasher	5	8
03-06-2055 19:00	2	HVAC	5	40
29-04-2058 12:00	2	Electronics	5	24
02-06-2089 11:00	2	Dishwasher	5	93
07-01-2101 02:00	2	Refrigerator	5	46
04-12-2120 04:00	2	Lighting	5	25
24-08-2120 12:00	2	HVAC	4.99	63
08-10-2112 06:00	2	Lighting	4.99	68
26-11-2101 09:00	2	HVAC	4.99	2
23-06-2099 14:00	2	Lighting	4.99	67
23-04-2095 14:00	2	Refrigerator	4.99	42
16-11-2097 18:00	2	Electronics	4.99	52

5) Data Loading Web App





6) Web Page with Dashboard

I. Energy Consumption by Appliance Trend

- HVAC contributes the most to energy consumption at 24.93%, followed by Lighting at 16.01%, and Washing Machine at 14.81%. The least energy is consumed by Electronics at 14.7%.
- Potential strategies to optimize such high consumption from some of the appliances include better insulation or smart lighting systems.

II. Energy Consumption by Day of the Week

- Energy consumption steadily decreases from Friday (highest) to Tuesday (lowest). Friday has the highest energy usage at 364K, while Tuesday has the lowest at around 358K.
- Possible explanations include reduced occupancy or altered activity patterns during weekdays versus weekends.

III. Occupancy Trend

- Occupied energy consumption (51.97%) is higher than unoccupied (48.03%). This indicates significant energy is consumed when the house is both occupied and unoccupied. Potential reasons for such high consumption even when it's unoccupied could likely be due to devices or appliances left running.
- Solutions to optimize the high consumption may include automated systems, such as smart plugs or energy-saving modes.

IV. Seasonal Trends

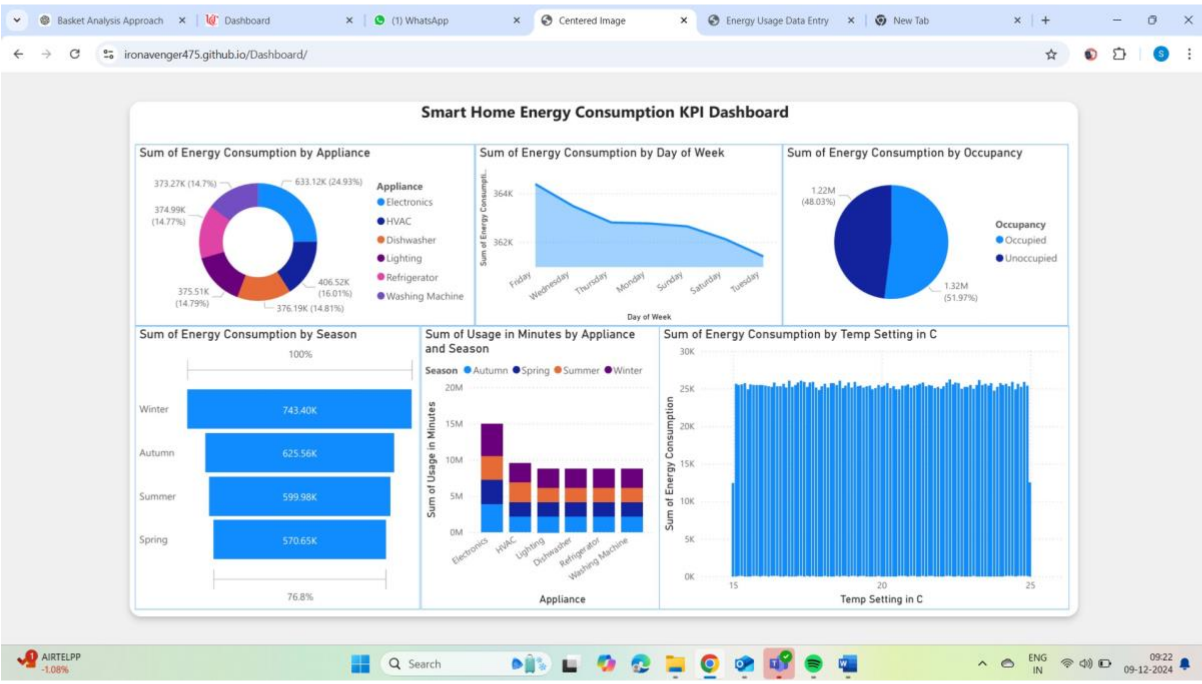
- Winter has the highest energy consumption (743.40K), while spring has the lowest (570.65K). Autumn and summer have similar values.
- What is the Season with the highest energy consumption?
  Winter.
- What contributes to the spike in winter energy consumption?
  It is clear that heating systems (HVAC) were being operated for longer hours during winter.

V. Energy Consumption by Temperature Setting

- Energy consumption remains relatively constant across temperature settings (15°C to 24°C).
- It is clear that the temperature setting has minimal to no impact in energy consumption.

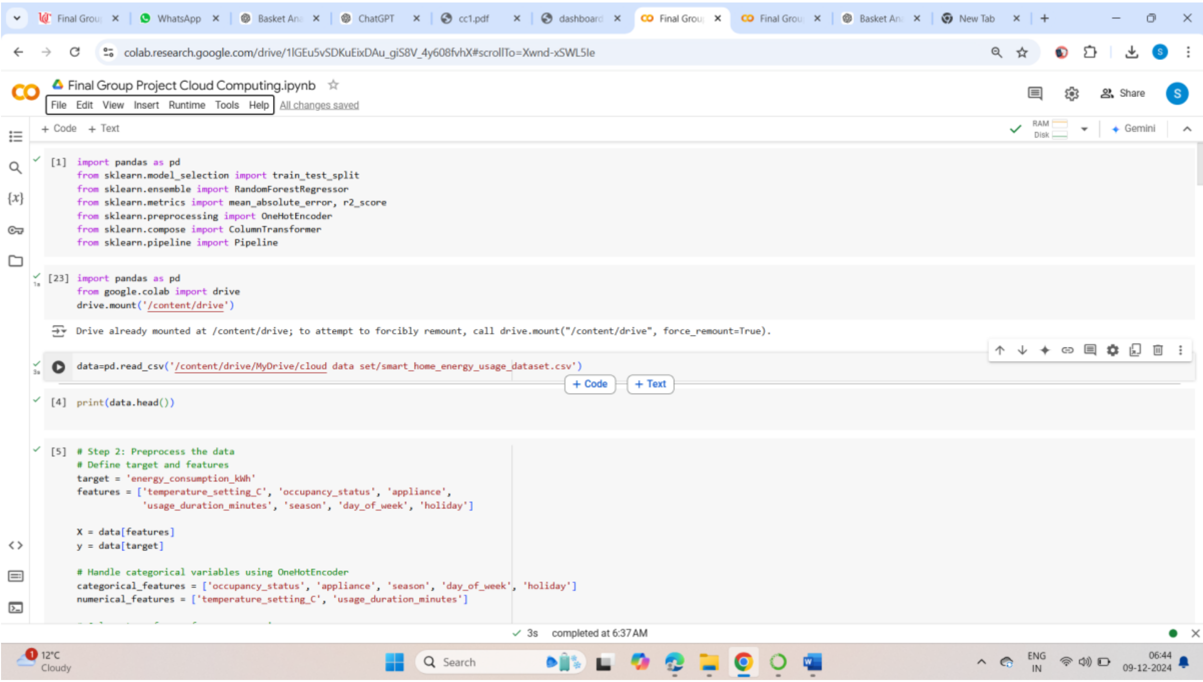
VI. Appliance Usage by Appliance and Season

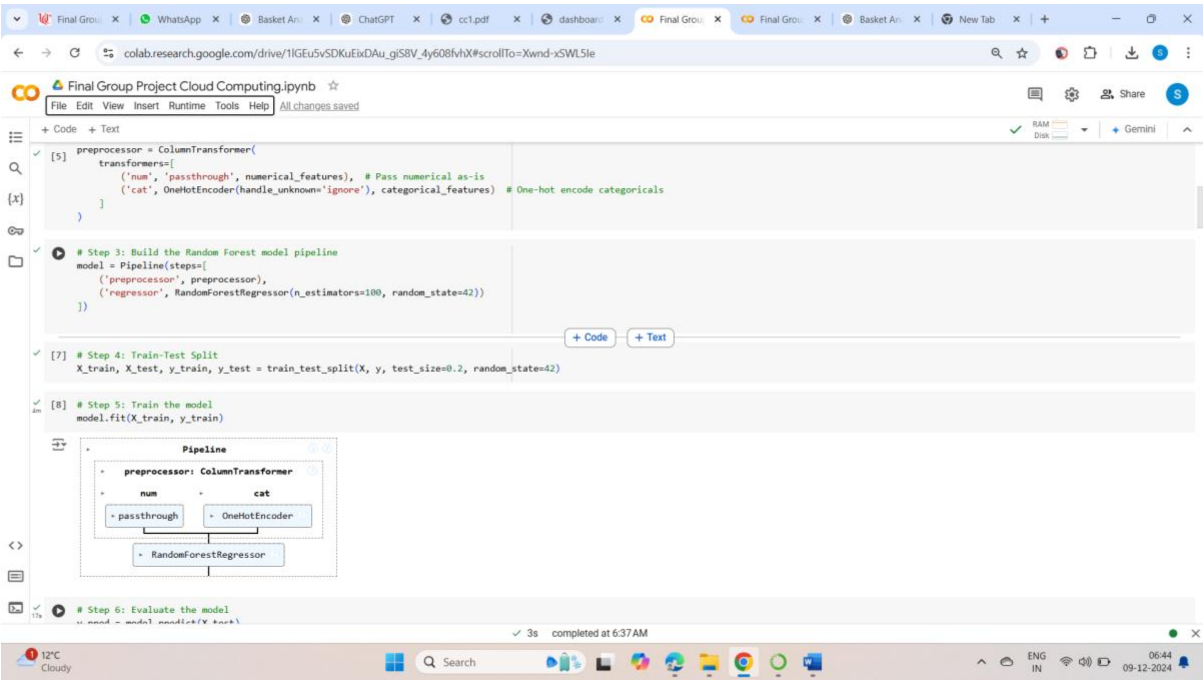
- Electronics dominate usage time across all seasons.
- HVAC has higher usage in winter compared to other seasons. Other appliances such as Lighting and Washing Machines have consistent usage across seasons.

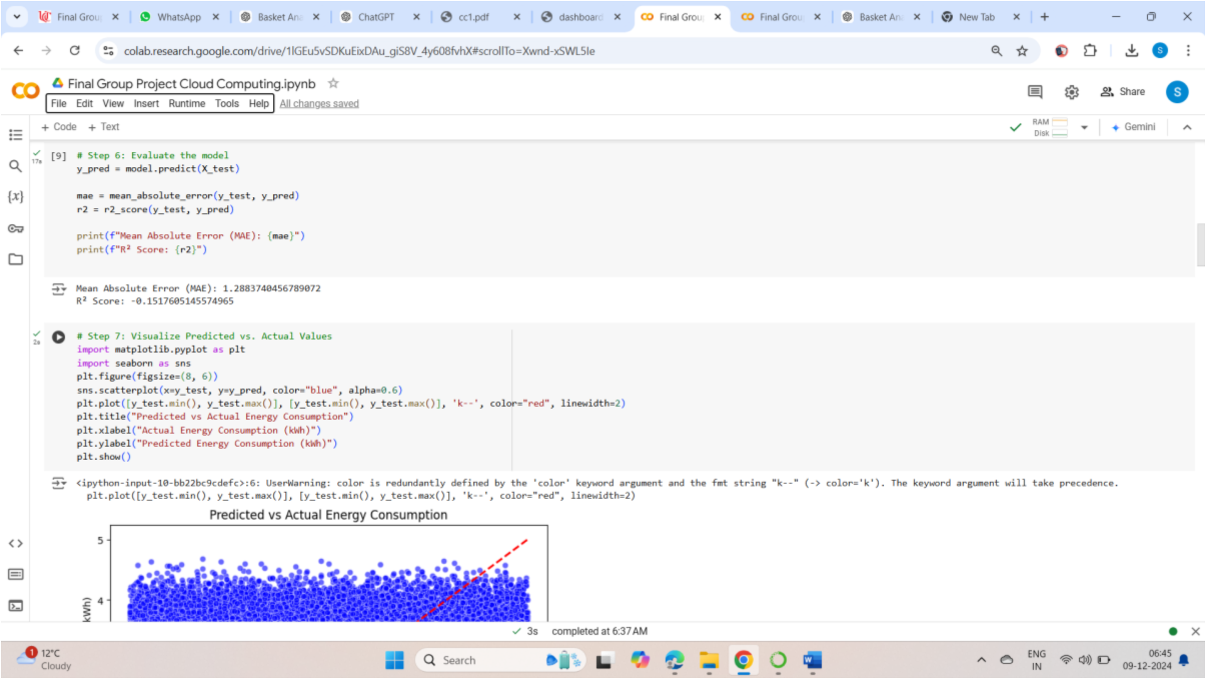


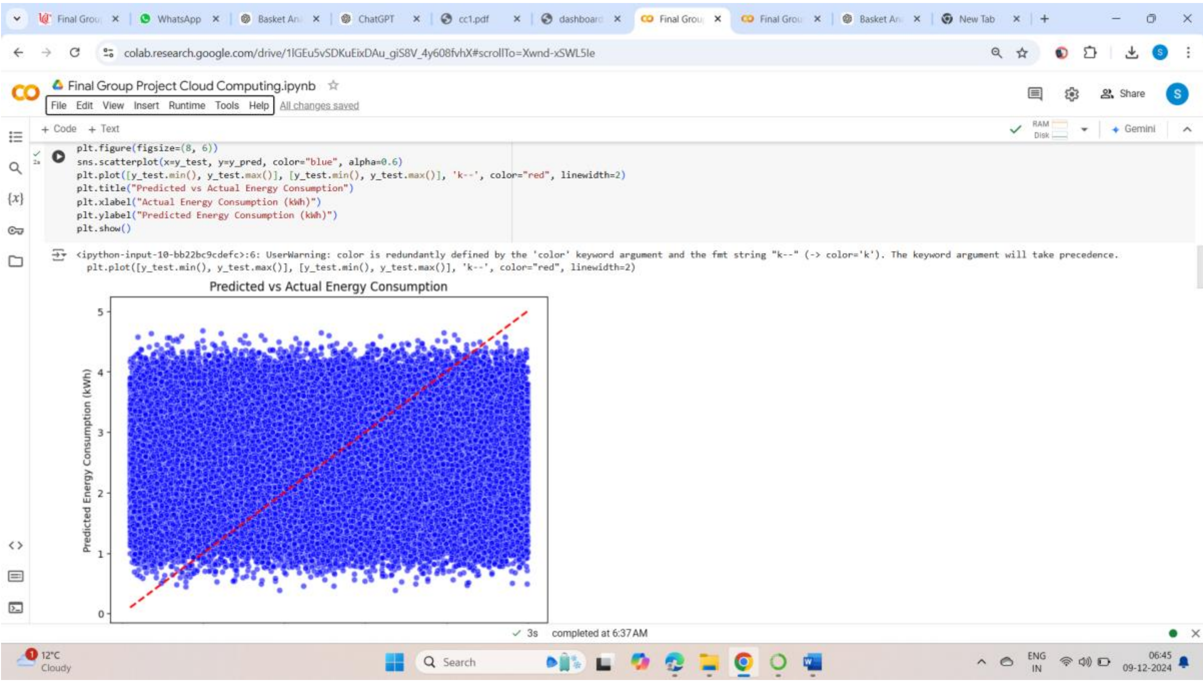
7) ML Model Application

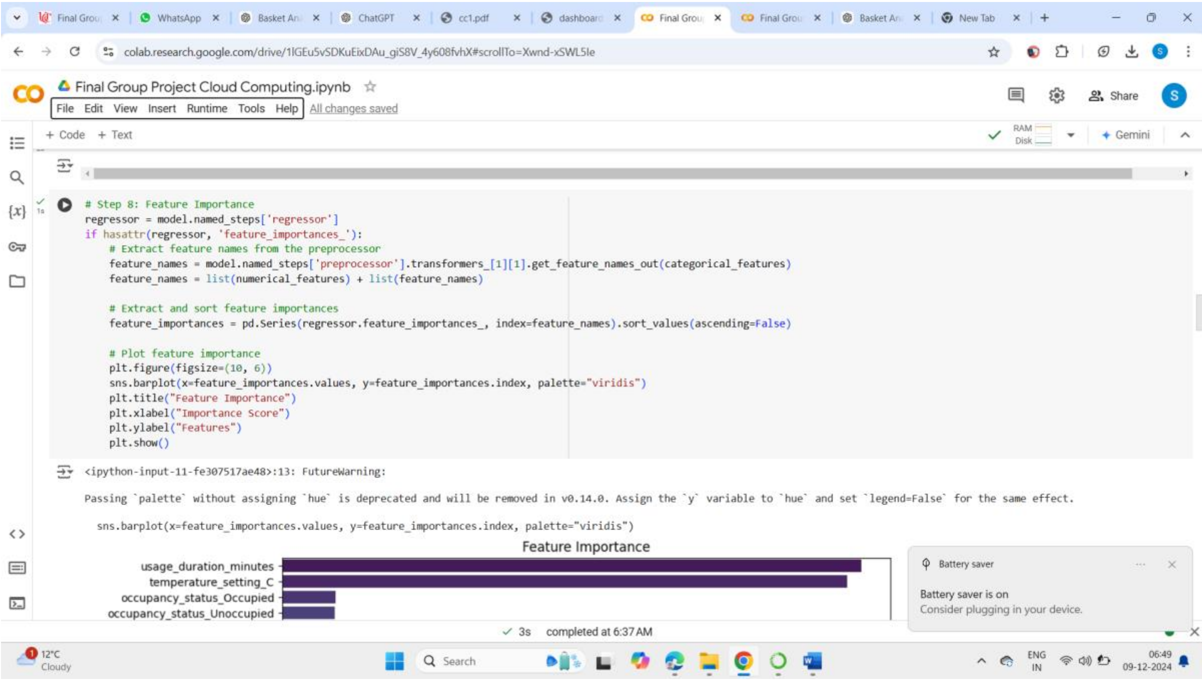
We used random forest for the basket analysis. Initially, we explored the dataset with a Random Forest Regressor to understand the relationship between the features (e.g., energy_consumption_kWh, temperature_setting_C, occupancy_status, appliance, etc.) and the target variable. The regressor helped identify key drivers of energy consumption, offering preliminary insights into feature importance and their influence on churn behavior.





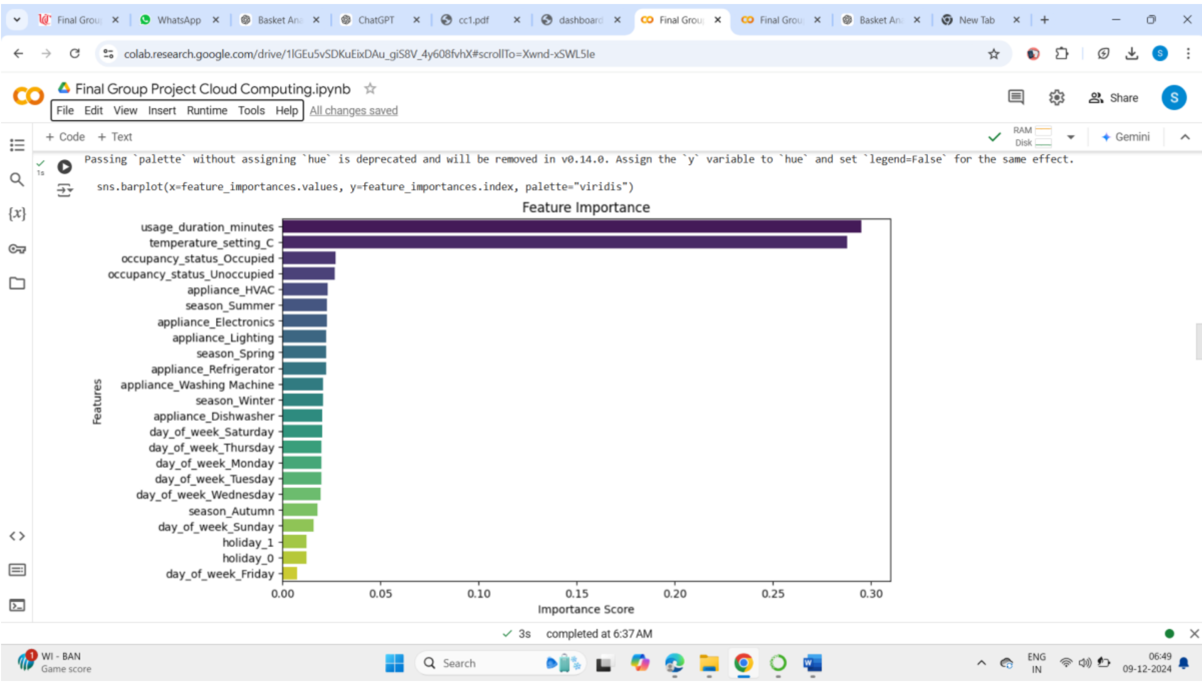

colab.research.google.com/drive/1lGEu5vSDKuEixDAu_giS8V_4y608fvhX#scrollTo=Xwnd-xSWL5le
Final Group Project Cloud Computing.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[9] # Step 6: Evaluate the model
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"R² Score: {r2}")
Mean Absolute Error (MAE): 1.2883740456789072
R² Score: -0.1517605145574965
# Step 7: Visualize Predicted vs. Actual Values
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, color="blue", alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', color="red", linewidth=2)
plt.title("Predicted vs Actual Energy Consumption")
plt.xlabel("Actual Energy Consumption (kWh)")
plt.ylabel("Predicted Energy Consumption (kWh)")
plt.show()
<ipython-input-10-bb22bc9cdefc>:6: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "k--" (-> color='k'). The keyword argument will take precedence.
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', color="red", linewidth=2)
Predicted vs Actual Energy Consumption
3s completed at 6:37 AM
12°C Cloudy
06:45 09-12-2024

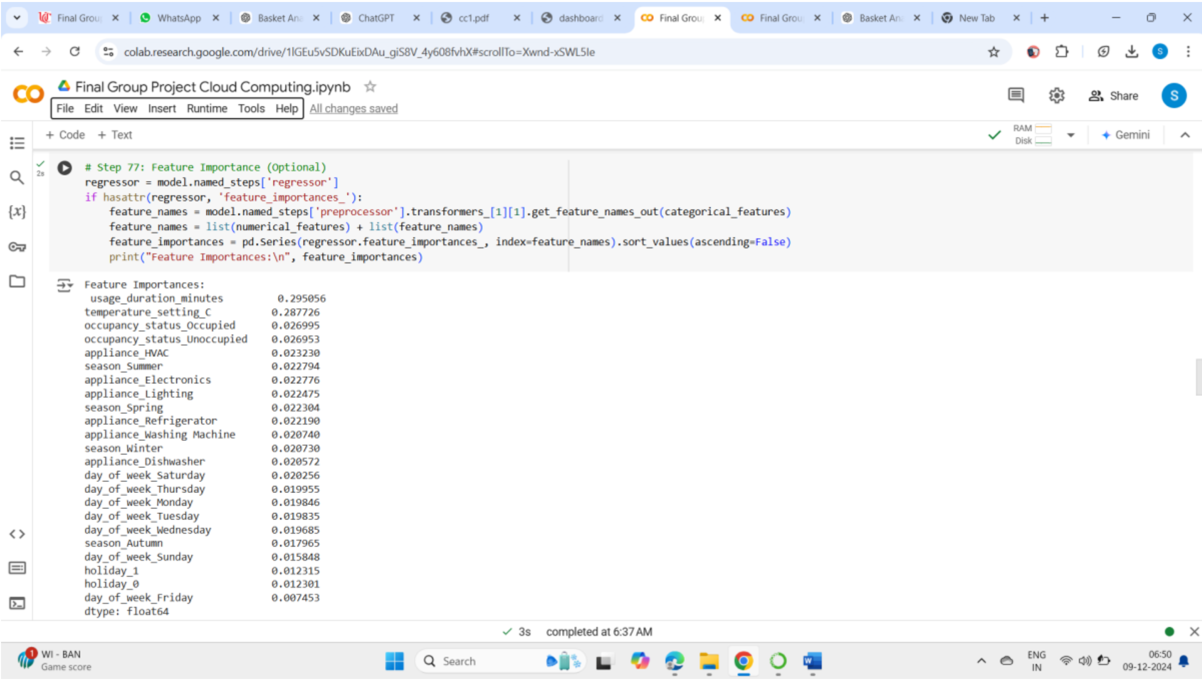

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, color="blue", alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', color="red", linewidth=2)
plt.title("Predicted vs Actual Energy Consumption")
plt.xlabel("Actual Energy Consumption (kWh)")
plt.ylabel("Predicted Energy Consumption (kWh)")
plt.show()
<ipython-input-10-bb22bc9cdefc>:6: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "k--" (-> color='k'). The keyword argument will take precedence.
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', color="red", linewidth=2)
Predicted vs Actual Energy Consumption
Predicted Energy Consumption (kWh)
0
1
2
3
4
5



```python
# Step 8: Feature Importance
regressor = model.named_steps['regressor']
if hasattr(regressor, 'feature_importances_'):
    # Extract feature names from the preprocessor
    feature_names = model.named_steps['preprocessor'].transformers_[1][1].get_feature_names_out(categorical_features)
    feature_names = list(numerical_features) + list(feature_names)

    # Extract and sort feature importances
    feature_importances = pd.Series(regressor.feature_importances_, index=feature_names).sort_values(ascending=False)

    # Plot feature importance
    plt.figure(figsize=(10, 6))
    sns.barplot(x=feature_importances.values, y=feature_importances.index, palette="viridis")
    plt.title("Feature Importance")
    plt.xlabel("Importance Score")
    plt.ylabel("Features")
    plt.show()
```

```
<ipython-input-11-fe307517ae48>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=feature_importances.values, y=feature_importances.index, palette="viridis")
```





8) Churn Prediction

In this project, we developed a churn prediction model using energy usage data from households. Churn was defined as a significant drop in energy consumption (energy_consumption_kWh) below 10% of the average usage. We selected relevant features such as temperature_setting_C, occupancy_status, appliance, and contextual factors like season, day_of_week, and holiday to predict churn behavior. Categorical features were one-hot encoded, while numerical features were passed directly into a preprocessing pipeline. A Random Forest Classifier was chosen due to its ability to handle non-linear relationships and provide insights into feature importance.

The dataset was split into training and testing subsets, and the model was trained and evaluated using metrics like precision, recall, F1-score, and a confusion matrix. Additionally, we visualized churn distribution and feature importance to better understand the key drivers of churn. This approach helps identify households likely to disengage, enabling businesses to design proactive strategies for customer retention based on energy consumption patterns and behavioral insights.

Final Group Project Cloud Computing.ipynb

```
[13] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import classification_report, confusion_matrix
     from sklearn.preprocessing import OneHotEncoder
     from sklearn.compose import ColumnTransformer
     from sklearn.pipeline import Pipeline
```

```
[14] # Step 2: Define the target (churn) variable
     # Example: Churn if energy consumption is less than 10% of mean
     threshold = data['energy_consumption_kWh'].mean() * 0.1
     data['churn'] = (data['energy_consumption_kWh'] < threshold).astype(int)
```

```
[15] # Step 3: Define features and target
     features = ['temperature_setting_C', 'occupancy_status', 'appliance',
                 'usage_duration_minutes', 'season', 'day_of_week', 'holiday']
     target = 'churn'

     X = data[features]
     y = data[target]
```

```
# Step 4: Preprocess data (handle categorical variables)
categorical_features = ['occupancy_status', 'appliance', 'season', 'day_of_week', 'holiday']
numerical_features = ['temperature_setting_C', 'usage_duration_minutes']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numerical_features),
```

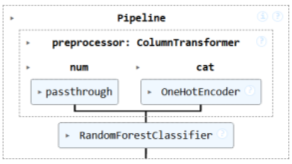
3s completed at 6:37 AM

Final Group Project Cloud Computing.ipynb

```
[17] # Step 5: Create a Random Forest pipeline
     model = Pipeline(steps=[
         ('preprocessor', preprocessor),
         ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
     ])
```

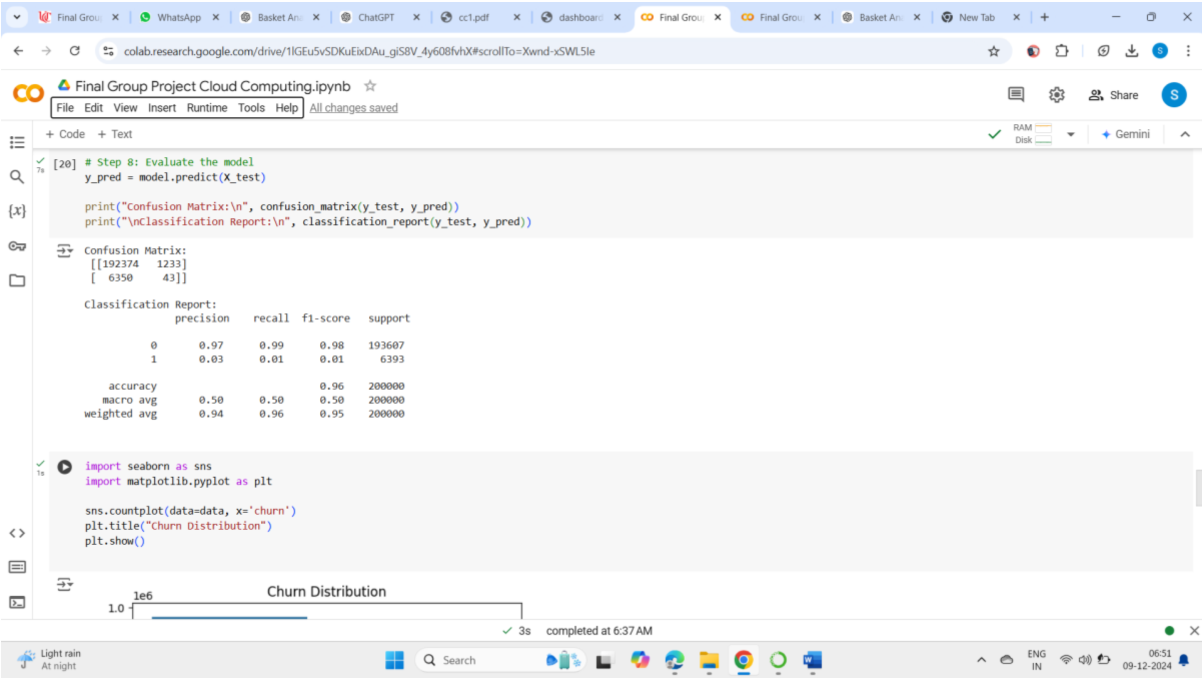
```
[18] # Step 6: Train-Test Split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

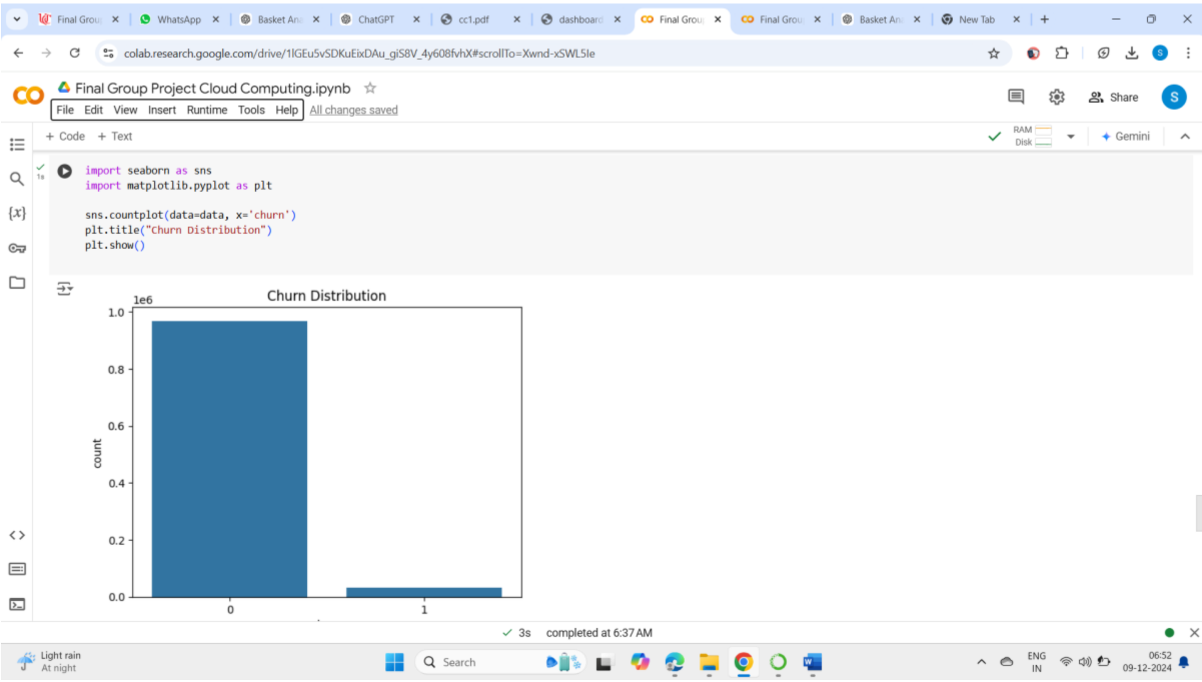
```
# Step 7: Train the model
model.fit(X_train, y_train)
```

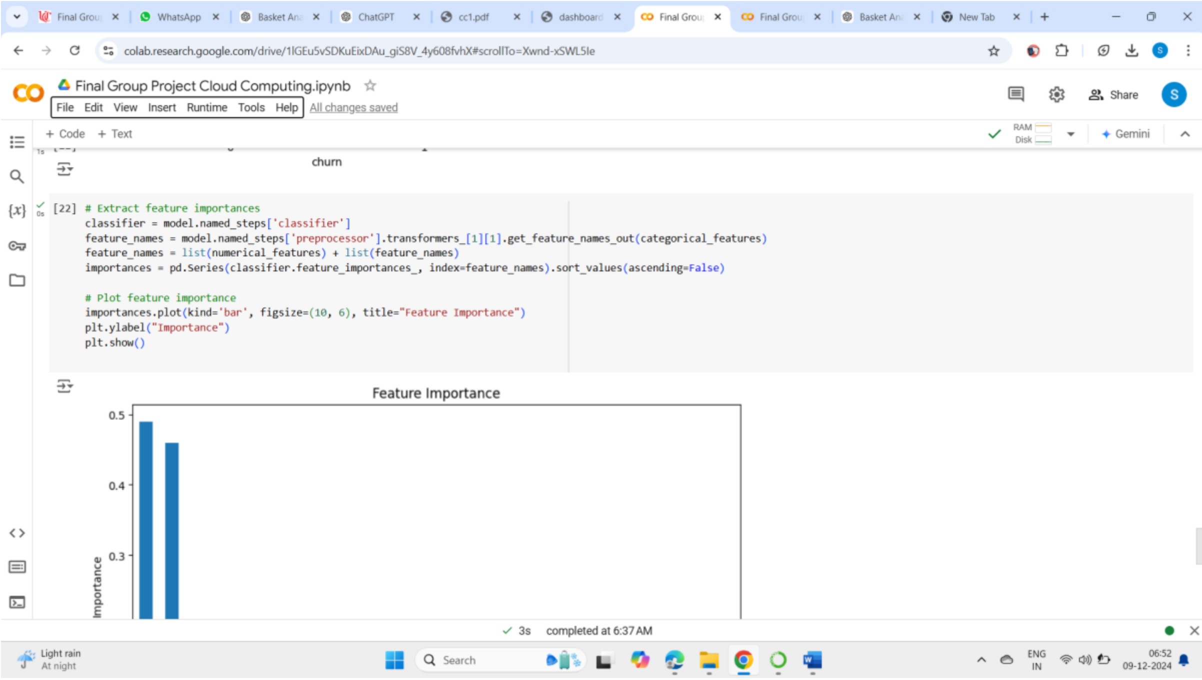


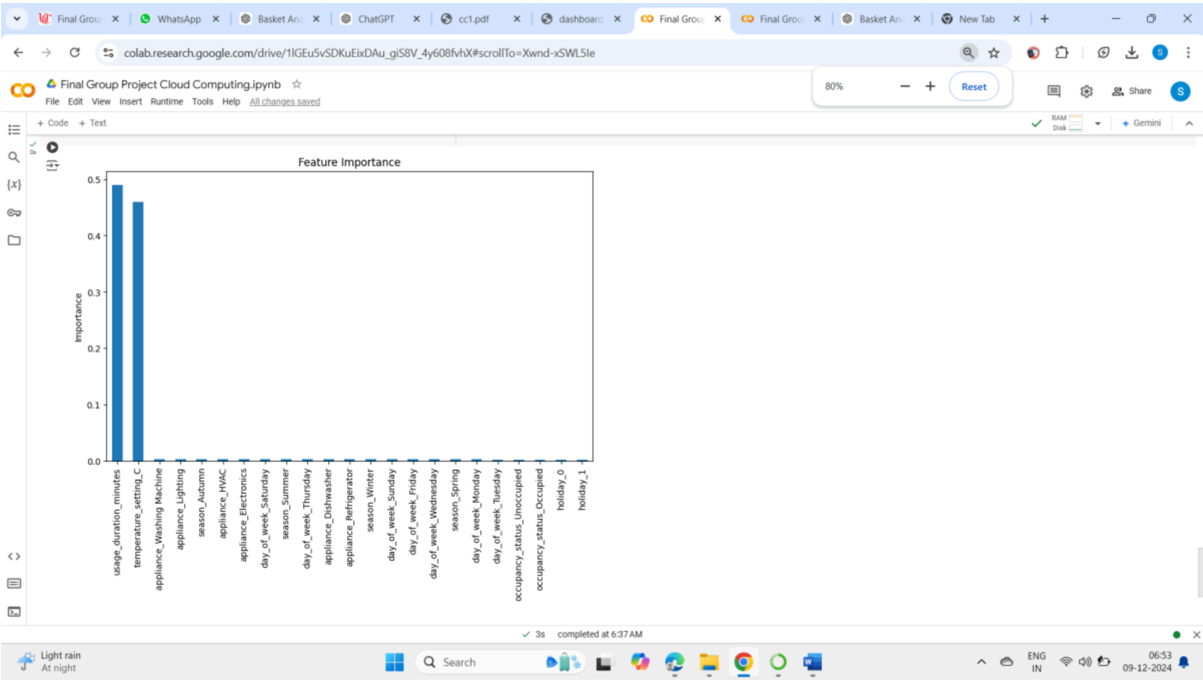

```
[20] # Step 8: Evaluate the model
     y_pred = model.predict(X_test)
```

3s completed at 6:37 AM


Final Group Project Cloud Computing.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[20] # Step 8: Evaluate the model
y_pred = model.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
Confusion Matrix:
[[192374 1233]
[6350 43]]
Classification Report:
precision recall f1-score support
0 0.97 0.99 0.98 193607
1 0.03 0.01 0.01 6393
accuracy 0.96 200000
macro avg 0.50 0.50 0.50 200000
weighted avg 0.94 0.96 0.95 200000
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(data=data, x='churn')
plt.title("Churn Distribution")
plt.show()
Churn Distribution
1e6
1.0
3s completed at 6:37 AM


Final Group Project Cloud Computing.ipynb
File Edit View Insert Runtime Tools Help All changes saved
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(data=data, x='churn')
plt.title("Churn Distribution")
plt.show()
Churn Distribution
1e6
1.0
0.8
0.6
0.4
0.2
0.0
count
0
1
3s completed at 6:37 AM


Final Group Project Cloud Computing.ipynb
File Edit View Insert Runtime Tools Help All changes saved
churn
[22]
# Extract feature importances
classifier = model.named_steps['classifier']
feature_names = model.named_steps['preprocessor'].transformers_[1][1].get_feature_names_out(categorical_features)
feature_names = list(numerical_features) + list(feature_names)
importances = pd.Series(classifier.feature_importances_, index=feature_names).sort_values(ascending=False)
# Plot feature importance
importances.plot(kind='bar', figsize=(10, 6), title="Feature Importance")
plt.ylabel("Importance")
plt.show()
Feature Importance
Importance
3s completed at 6:37 AM


Final Group Project Cloud Computing.ipynb
Feature Importance
Importance
0.0
0.1
0.2
0.3
0.4
0.5
usage_duration_minutes
temperature_setting_C
appliance_Washing Machine
appliance_Lighting
season_Autumn
appliance_HVAC
appliance_Electronics
day_of_week_Saturday
season_Summer
day_of_week_Thursday
appliance_Dishwasher
appliance_Refrigerator
season_Winter
day_of_week_Sunday
day_of_week_Friday
day_of_week_Wednesday
season_Spring
day_of_week_Monday
day_of_week_Tuesday
occupancy_status_Unoccupied
occupancy_status_Occupied
holiday_0
holiday_1
3s completed at 6:37 AM