Explanation:

MODEL SELECTION:

Among the models that I knew,I chose Randomforest over others(like LinearRegression and DecisionTrees)because:

1)Patterns largely depend on the Venue_ID..Therefore using a model like Linear Regression would not give accurate predictions.

2)Since making different models for each venue is not much suggestable and it creates unneccesary chaos.
I decided to make use of DecisionTrees..

3)Since the features are diverse enough..
We need to overcome the overfitting that a single DecisionTree might cause..
4)Therefore keeping this in mind I decided to go with RANDOMFOREST.

HyperParameter Tuning:

1.***GridSearch:*** I used Grid Search because it is an exhaustive search method. It guarantees that, within the range of values we specified, we will find the best combination of hyperparameters, rather than guessing randomly.

2.VaLiDaTiOn StRaTeGy FoR TuNiNg :

K-Fold Cross-Validation: I used K-Fold Cross-Validation to ensure my model's performance wasn't a fluke. By testing the model on different subsets of the data, I proved that it generalizes well to new data and isn't just memorizing one specific training set.

3.THE PARAMETERS THAT I TUNED:

1)***n-estimators:*** I tuned n_estimators to find the balance between accuracy and computational cost. Generally, more trees lead to more stable predictions (better averaging), but adding too many slows down the model without adding much value.

2)***max_depth:*** I tuned max_depth to prevent overfitting. By limiting how deep the trees can grow, we force the model to learn general rules rather than memorizing every specific detail of the training data.

3)***min_samples_split & min_samples_leaf:*** min_samples_split is the minimum number of samples that are required inorder to make a new rule in out decisionTree..
min_samples_leaf is the minimum number of leaves that are needed inorder to make a final prediction..

I tuned these inorder to handle OVERFITTING..

DOCUMENTATION OF HYPERPARAMETERS EXPLORED:

I tested a total of 81 different model combinations (3 × 3 × 3 × 3) to identify the optimal configuration.

- **n_estimators (Number of Trees)**
  - Values Tested: [100, 200, 300]
  - Reasoning:I tested a range from 100 to 300 to observe if increasing the number of "voters" (trees) improved model stability and accuracy, or if a smaller number (100) was sufficient to capture the data patterns without unnecessary computational cost.
- **max_depth (Maximum Tree Depth)**
  - Values Tested: [10, 20, None]
  - Reasoning:

10 & 20: I tested restricted depths to force the model to learn general trends and prevent it from memorizing specific noisy data points (overfitting).

None: I also included None to allow the trees to grow until all leaves were pure, providing a baseline to see if fully complex trees performed better than restricted ones.

- **min_samples_split (Minimum Samples to Split)**
  - Values Tested: [2, 5, 10]
  - Reasoning: I explored increasing this value from the default (2) to 5 and 10. A higher value (e.g., 10) acts as a "brake," preventing the model from creating very specific rules based on tiny groups of concerts, which helps reduce overfitting.
- **min_samples_leaf (Minimum Samples at Leaf)**
  - Values Tested: [1, 2, 4]
  - Reasoning: By testing values greater than 1 (like 2 and 4), I ensured that every final energy prediction was based on at least a few concerts, rather than a single outlier event. This improves reliability.

Final hyperparameter values with reasoning for selection:

After running Grid Search with Cross-
Validation, I found the best settings for my Random Forest model were:

n_estimators: 300

max_depth: 10

min_samples_split: 10

min_samples_leaf: 2

Reasoning for Selection: The Grid Search algorithm picked this specific combination because it gave the highest accuracy on the validation data. Here is why I think these values worked best:

For n_estimators (300): I tested 100, 200, and 300. The model selected the highest value, **300**. This indicates that the dataset has enough complexity that adding more trees continued to improve the stability of the predictions. By averaging the results of 300 "voters" (trees) instead of just 100 or 200, the model was able to smooth out errors and provide a more accurate final score.

For max_depth (10): I observed that the model preferred a depth of 10 (the lowest value I tested) over 20 or None. This is a clear sign that the deeper trees were "overfitting" (memorizing noise). By sticking to 10, I forced the model to keep things simple and look for general rules instead of specific outliers.

For min_samples_split (10): The model chose a higher value (10) instead of the default (2). This acts like a filter; it tells the model, "Don't create a new decision branch unless you have at least 10 concerts to look at." This helped remove noise and made the model more reliable.

For min_samples_leaf (2): The model preferred a value of **2** rather than the default 1. This acts as a safety filter against outliers. By requiring at least 2 concerts to land in a final leaf.

Comparison of tuned model vs. default parameters:

To understand if my hyperparameter tuning actually helped, I compared the accuracy (R2 Score) of the Tuned Model against the Default Model.

Default Model R2 Score: 0.5801

Tuned Model R2 Score: 0.6283

There is a 5% Increase in the accuracy!

I feel that the biggest difference lies in the Tree Depth and Splitting Rules.

I think the default model was overfitting since there is no limitation on the tree depth...But the tuned model has a limitation of 10 which forces it to make general rules applying for a wide range of samples.

Therefore what I witness here is that "more complex" is not always better..

NOTE:

I have tried flooring the rows with Crowd_Energy values=0 to a random small value between 3 to 15...I was getting a model with accuracy around 56%.

Feature Engineering: I have also tried to implement interaction between the features Volume_Level and Crowd_Energy and club them both into a single column named Volume_Density..

Despite having a good correlation value it decreased the model efficiency when i used..So I am not mentioning it overhere.

I controlled myself to use only Thresholds and Transformations at required places and got good results from them...

KEY FINDINGS FOR EACH VENUE:

- **V_Alpha:**
    - **Volume_Level: Exhibits a sound limit; higher volumes do not continuously increase crowd energy beyond a certain point, with a correlation of 0.185.**
    - **Crowd_Size: Positively correlates with crowd energy (0.185).**
    - **Weather: Stormy weather negatively impacts crowd energy.**
- **V_Beta:**
    - **Time_Label: Late Night shows higher crowd energy; overall, V_Beta performs better at night.**
    - **Weather: Stormy weather negatively affects crowd energy.**
    - **Day_of_Week: Matters significantly, causing fluctuations in crowd energy.**
- **V_Delta:**
    - **Volume_Level: Shows a strong positive correlation, indicating a high dependency on volume.**

- **Crowd_Size: Increases with crowd size (correlation of 0.164).**
- **Ticket_Price: Is negatively affected by an increase in ticket price (correlation of -0.344).**
- **Weather: Stormy weather negatively affects crowd energy.**

- **V_Gamma:**
  - **Ticket_Price: Shows a unique and drastic increase in crowd energy with price, exhibiting a strong positive correlation of 0.443.**
  - **Time_Label: Afternoons are particularly better for crowd energy.**
  - **Opener_Rating: Matters the most among all factors, showing the highest positive correlation (0.472) across all venues.**

Singer's INCORRECT Theories:

1) Tuesdays are not cursed..It is just a myth of the Singer.
2) Moon_Phase doesn't have any affect on the Crowd_Energy.
3) Band_Outfit also doesn't influence Crowd_Energy.

Note: In depth EDA and conclusions were mentioned along with the code in the notebook markdowns.