Possible Implementation

Here are the possible implementation we can choose to do in order to improve the reliability of the Project.

1. Matrices based solution , instead of single list based solution.

Currently, I am passing two string to a model and the model is calculating the similarity score based on the feature and giving the result.
One another approach will be sending a matrices based on single hotel, Thus instead of for loop with the references and the supplier data, two matrices of references and supplier will be sent to the model for the similarity calculation.

It is hard to assume, one to one relationship will be found in this case. we can use Hungarian matching algorithm over the similarity map and output the one to one relationship.

2. Strong embedding model for feature generation

I have only tried to create embedding based on : all-MiniLM-L6-v2
There are many other models which might performs better. To be honest, the keywords used in the domain is very less and thus, A fine tuned embedding model might work better even it is over-fitted in nature.

when trying embedding model, many dimensionality reduction approach can be used like PCA/UMAP as string length of room name is small and a smaller model is being trained at the end.

2.1 Try TabTransformer

There are few transformer variant which works well with the tabular data. The model works well with the categorical data as well, like room size, room type, smoking_status, pet_status features.

3. Adding more room feature to the data.

My usual answer, more feature might help the model, but we need to check feature importance.

3.1 Handling categorical and numerical data well.

Rightnow, I am considering all the feature as numerical data and no new categorical data is added. but once we add more feature the system, this hybrid data format should handle well.

```
from sklearn.compose import ColumnTransformer

preprocessor = ColumnTransformer(
    [
        ("one-hot-encoder", categorical_preprocessor,
categorical_columns),
        ("standard_scaler", numerical_preprocessor, numerical_columns),
    ]
)
```

something like this.

4. Trying different models and improve Hyper parameter search

Optuna is a great tool to do the hyper-parameter search.
But it is slower to check every possible pattern and sometimes it is hard to reason the findings.
model_tuner is implemented with the system, yet, I haven't looked deep inside of it.
It is also useful, if we have multiple model like xgboost, lightbgm, logistic regression and using the hyper parameter would also make sense.

5. Hybrid Inference Architecture

Having debug flag is nice to debug what it is. Alongside of that, we can add flag like `deepcheck: True` which enable to use multiple model to do strong check for the room matching.
This may include active search in the supplier platform (using API) fetch fresh data and generate the matching.