Airbnb Price Prediction Model

Chukwudi obianwu victor
University Of Victoria
Chukwudio@uvic.ca

Introduction/Overview

- Welcome to a groundbreaking exploration of Airbnb price prediction, where tradition meets innovation. In a landscape dominated by conventional models, we dare to ask: Can Large language models redefine accuracy in predicting short-term accommodation prices? This presentation unfolds the narrative of how the Large Language Model (LLM) disrupts the status quo, challenging the established norms of predictive analytics.

- In the evolving world we find ourselves in as of today, it is important to consider better means and modern approaches such as Large Language models to predicting Airbnb prices.

Objectives/Motivation

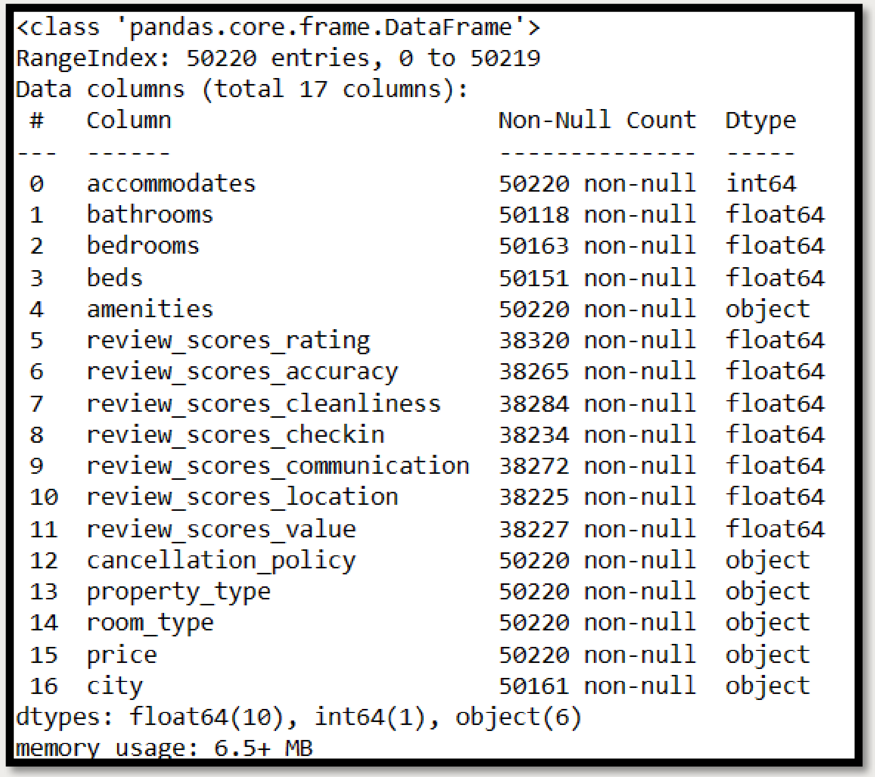
This project is guided by two core objectives:

- Benchmarking Accuracy: Compare the accuracy of the Language Model (LLM) against traditional models such as linear regression, gradient boost, and more.
- Democratizing Predictions: Showcase that the LLM, known for its language understanding prowess, can achieve comparable accuracy to traditional techniques without the complexity.

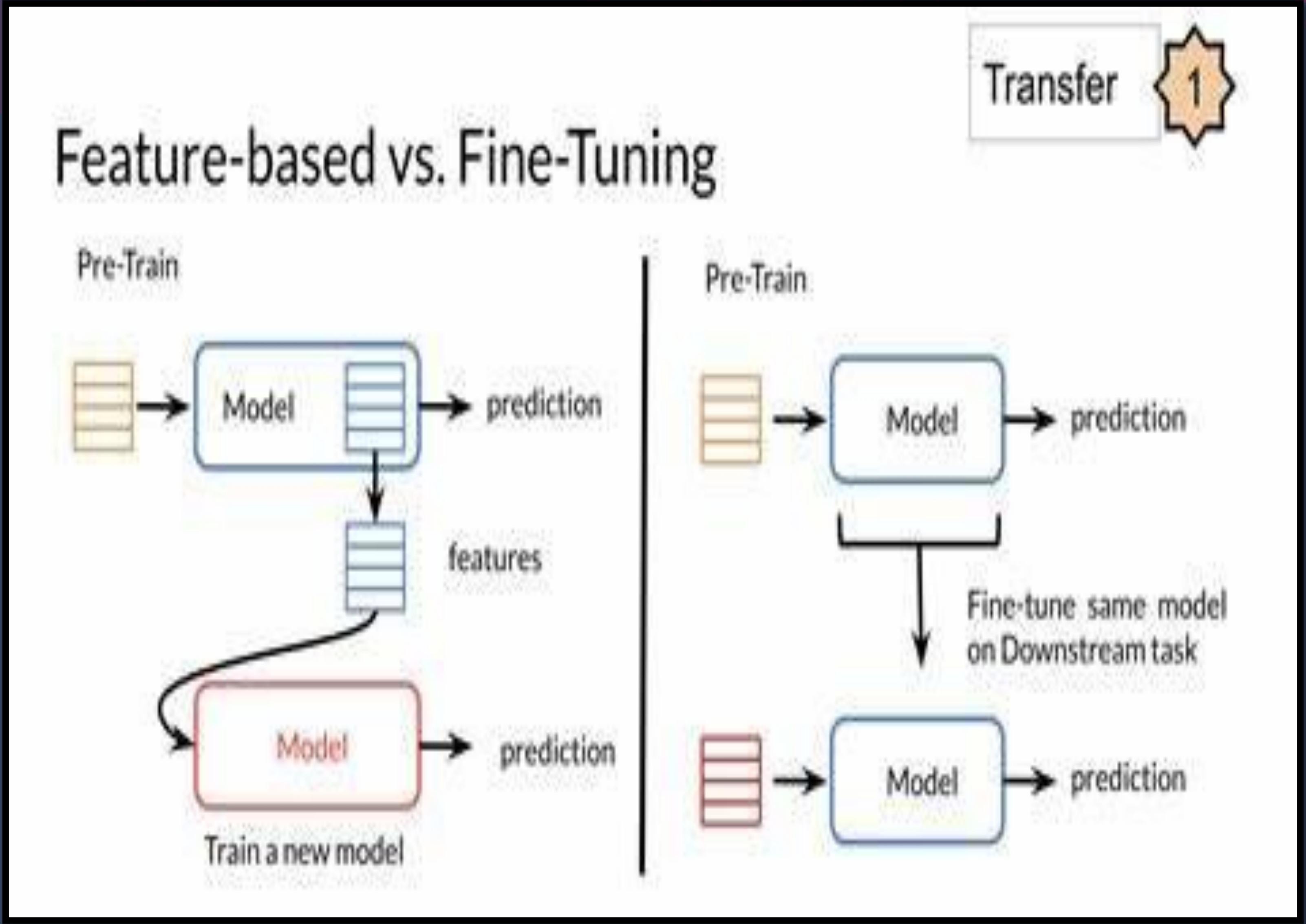
The motivation behind this project lies in questioning the assumption that intricate machine learning models are the sole path to accurate predictions. By leveraging the capabilities of the Language Model, we aim to demonstrate that predicting Airbnb prices doesn't require complex algorithms, but rather an understanding of language nuances.

Dataset

- The dataset used was the public Airbnb dataset for New York city. This dataset consists of 50,221 entries and 96 features. This dataset was similarly used by a set previous of developers to compare model accuracies in Airbnb price prediction which used traditional approaches in the development of each model.
- The dataset comprises of both categorical and numerical features
- The dataset also had a few columns with missing values.
- Due to the cost per token, only the features which were deemed to be relevant were selected from the dataset to be used in training and validation.
- The image at the side consists of the relevant features selected for the pre-processing and their data types.



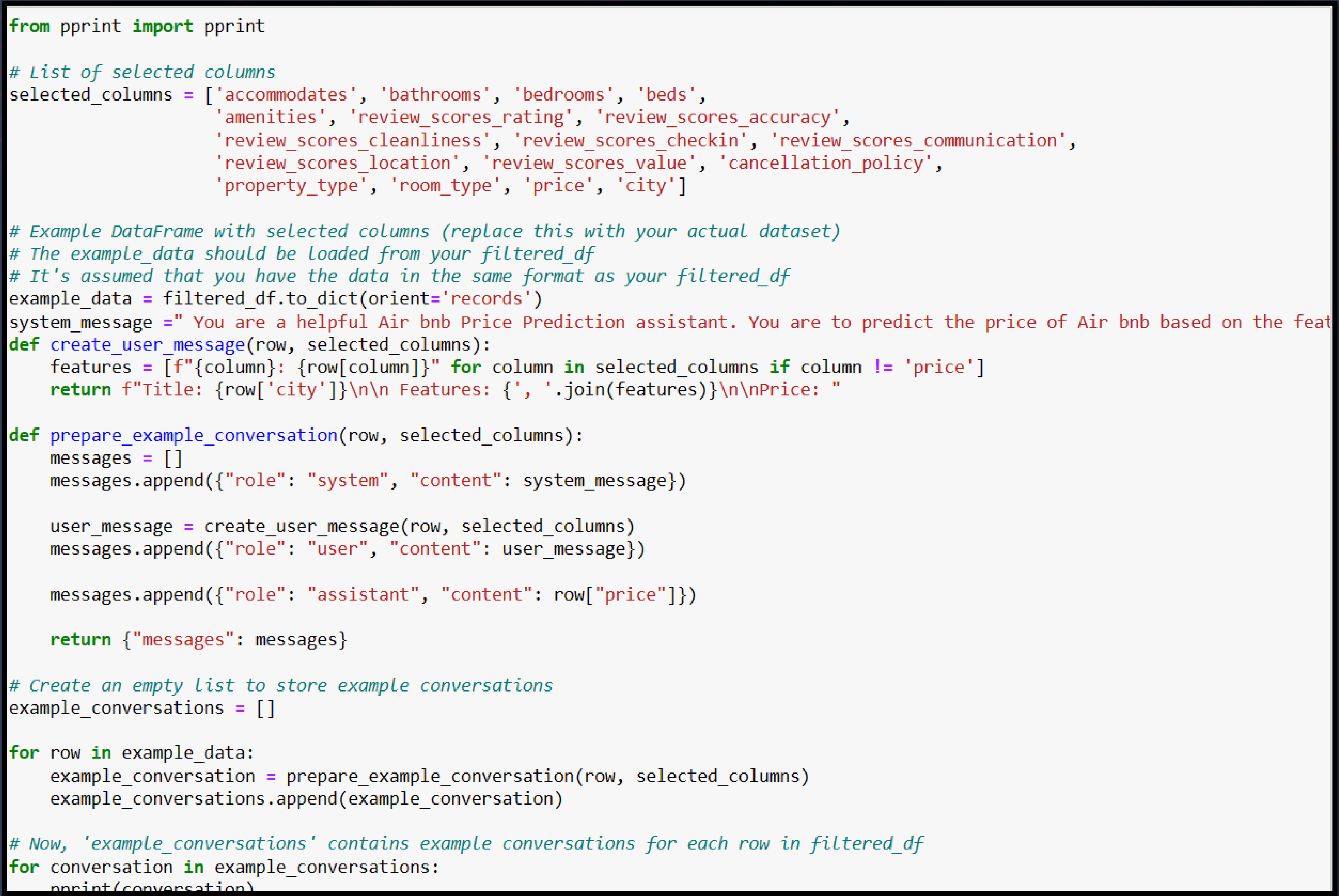
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50220 entries, 0 to 50219
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   accommodates                 50220 non-null  int64
 1   bathrooms                    50118 non-null  float64
 2   bedrooms                     50163 non-null  float64
 3   beds                         50151 non-null  float64
 4   amenities                    50220 non-null  object
 5   review_scores_rating         38320 non-null  float64
 6   review_scores_accuracy       38265 non-null  float64
 7   review_scores_cleanliness    38284 non-null  float64
 8   review_scores_checkin        38234 non-null  float64
 9   review_scores_communication  38272 non-null  float64
 10  review_scores_location       38225 non-null  float64
 11  review_scores_value          38227 non-null  float64
 12  cancellation_policy          50220 non-null  object
 13  property_type                50220 non-null  object
 14  room_type                    50220 non-null  object
 15  price                        50220 non-null  object
 16  city                         50161 non-null  object
dtypes: float64(10), int64(1), object(6)
memory usage: 6.5+ MB
```


Transfer
1
Feature-based vs. Fine-Tuning
Pre-Train
Model
prediction
features
Model
prediction
Train a new model
Pre-Train
Model
prediction
Fine-tune same model on Downstream task
Model
prediction

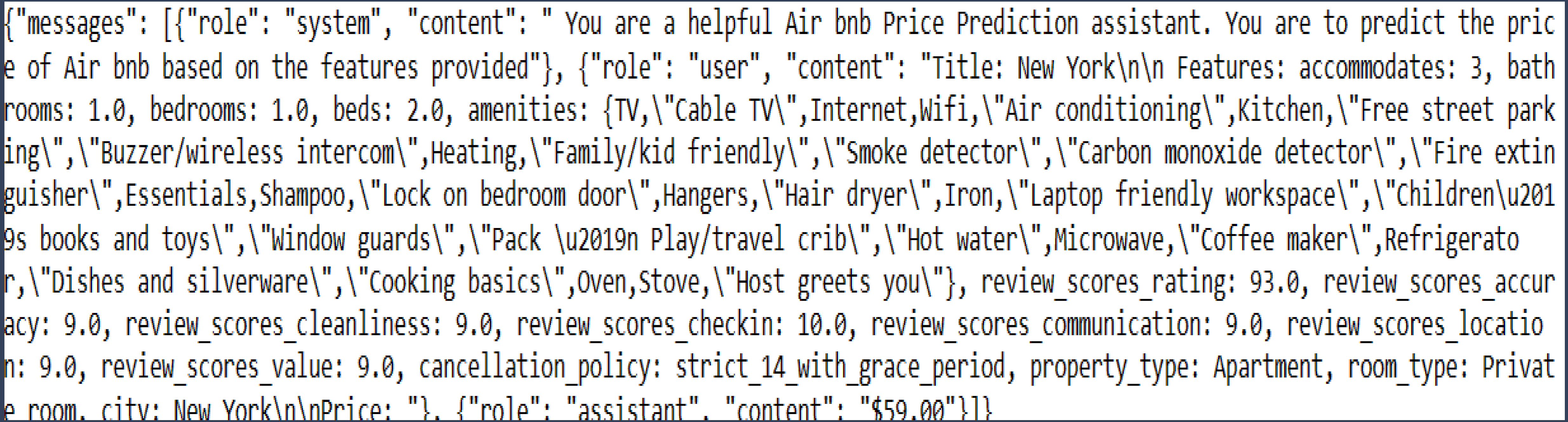
Data Preparation

For the data preparation the following stages were gone through;

- Data Representation: In this stage the training samples are represented in a list of message each having different roles, one for system, one for user and one for assistant. The system message sets the context for the assistant, defining its purpose while the user message contains the input features(e.g ,accomaodation, city). And the expected price. While the assistant message contains the models predictive price.
- Data selection: A subsect of relevant columns are selected from all the features in the dataset and the rest were dropped.
- Data Transformation: the selected columns were converted into a dictionary format to match the expected input for fine tuning.
- Training and validation: The first 100 rows were selected for training and rows 101- 200 were selected for validation due to the cost per token.
- Json File Creation: the formatted training and validated files(represented as list of messages) were saved in json file format
- File Inspection: Then the first the first 5 lines of the training and validation jsonl files was displayed to ensure correctness in the files format.



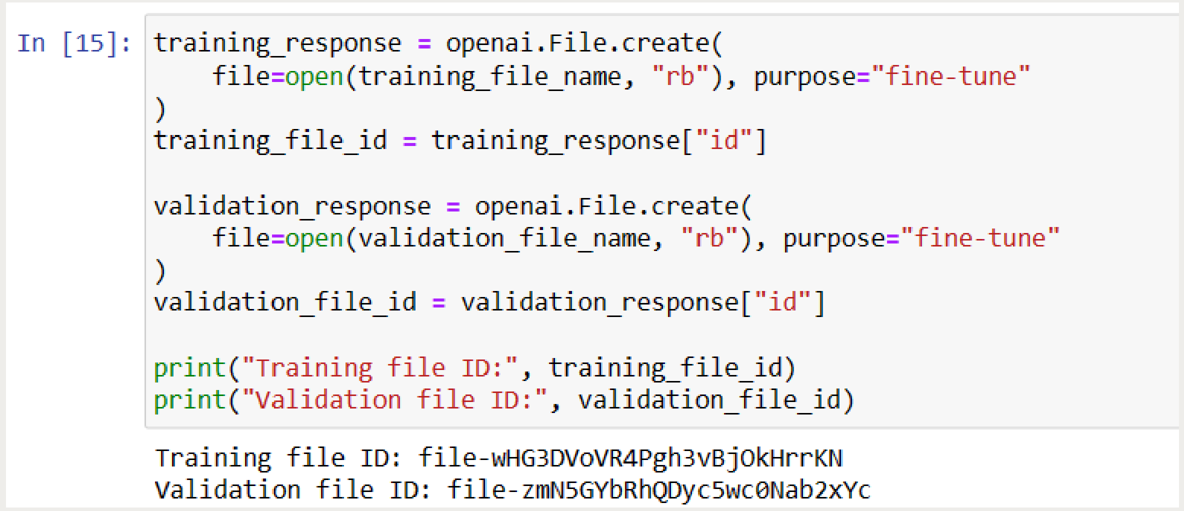
```python
from pprint import pprint

# List of selected columns
selected_columns = ['accommodates', 'bathrooms', 'bedrooms', 'beds',
                    'amenities', 'review_scores_rating', 'review_scores_accuracy',
                    'review_scores_cleanliness', 'review_scores_checkin', 'review_scores_communication',
                    'review_scores_location', 'review_scores_value', 'cancellation_policy',
                    'property_type', 'room_type', 'price', 'city']

# Example DataFrame with selected columns (replace this with your actual dataset)
# The example_data should be loaded from your filtered_df
# It's assumed that you have the data in the same format as your filtered_df
example_data = filtered_df.to_dict(orient='records')
system_message =" You are a helpful Air bnb Price Prediction assistant. You are to predict the price of Air bnb based on the feat
def create_user_message(row, selected_columns):
    features = [f"{column}: {row[column]}" for column in selected_columns if column != 'price']
    return f"Title: {row['city']}\n\n Features: {', '.join(features)}\n\nPrice: "

def prepare_example_conversation(row, selected_columns):
    messages = []
    messages.append({"role": "system", "content": system_message})

    user_message = create_user_message(row, selected_columns)
    messages.append({"role": "user", "content": user_message})

    messages.append({"role": "assistant", "content": row["price"]})

    return {"messages": messages}

# Create an empty list to store example conversations
example_conversations = []

for row in example_data:
    example_conversation = prepare_example_conversation(row, selected_columns)
    example_conversations.append(example_conversation)

# Now, 'example_conversations' contains example conversations for each row in filtered_df
for conversation in example_conversations:
    pprint(conversation)
```



```
{"messages": [{"role": "system", "content": " You are a helpful Air bnb Price Prediction assistant. You are to predict the pric
e of Air bnb based on the features provided"}, {"role": "user", "content": "Title: New York\n\n Features: accommodates: 3, bath
rooms: 1.0, bedrooms: 1.0, beds: 2.0, amenities: {TV,\"Cable TV\",Internet,Wifi,\"Air conditioning\",Kitchen,\"Free street park
ing\",\"Buzzer/wireless intercom\",Heating,\"Family/kid friendly\",\"Smoke detector\",\"Carbon monoxide detector\",\"Fire extin
guisher\",Essentials,Shampoo,\"Lock on bedroom door\",Hangers,\"Hair dryer\",Iron,\"Laptop friendly workspace\",\"Children\u201
9s books and toys\",\"Window guards\",\"Pack \u2019n Play/travel crib\",\"Hot water\",Microwave,\"Coffee maker\",Refrigerato
r,\"Dishes and silverware\",\"Cooking basics\",Oven,Stove,\"Host greets you\"}, review_scores_rating: 93.0, review_scores_accur
acy: 9.0, review_scores_cleanliness: 9.0, review_scores_checkin: 10.0, review_scores_communication: 9.0, review_scores_locatio
n: 9.0, review_scores_value: 9.0, cancellation_policy: strict_14_with_grace_period, property_type: Apartment, room_type: Privat
e room, city: New York\n\nPrice: "}, {"role": "assistant", "content": "$59.00"}]}
```

Uploading

- In the preprocessing stage, the preprocessed data files are uploaded to the OpenAI Files endpoint for subsequent fine-tuning of the ChatGPT model. This involves two key steps:
- First, the training data file (training Jsonl file) is uploaded using the openai.File.create method with the purpose specified as "fine-tune." The file ID (training_file_id) is extracted from the response for reference.
- Second, a similar process is followed for the validation data file (validation_file_name), and its file ID (validation_file_id) is obtained. The display of both training and validation file IDs in the console ensures the successful upload of the files, making them ready for the fine-tuning process.



```
In [15]: training_response = openai.File.create(
             file=open(training_file_name, "rb"), purpose="fine-tune"
         )
         training_file_id = training_response["id"]

         validation_response = openai.File.create(
             file=open(validation_file_name, "rb"), purpose="fine-tune"
         )
         validation_file_id = validation_response["id"]

         print("Training file ID:", training_file_id)
         print("Validation file ID:", validation_file_id)

         Training file ID: file-wHG3DVoVR4Pgh3vBjOkHrrKN
         Validation file ID: file-zmN5GYbRhQDyc5wc0Nab2xYc
```

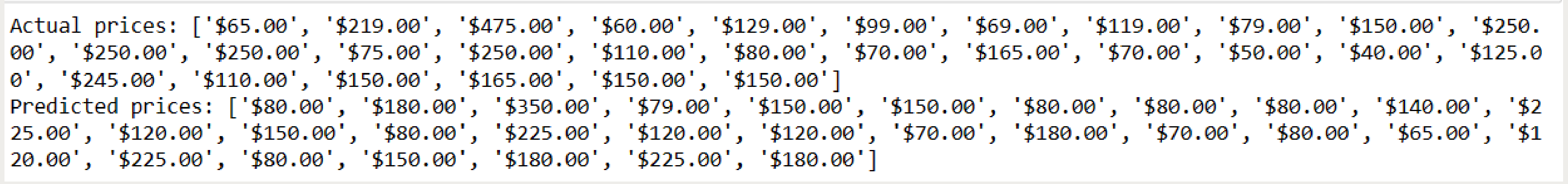
Fine Tuning/ Inference

Fine Tuning

In the fine-tuning process, the ChatGPT model is customized using prepared training and validation data files. Key steps include initiating the fine-tuning job, monitoring progress through status retrieval, and tracking events like training and validation loss. Upon successful completion, the fine-tuned model is created, and its ID is extracted for reference

Inference

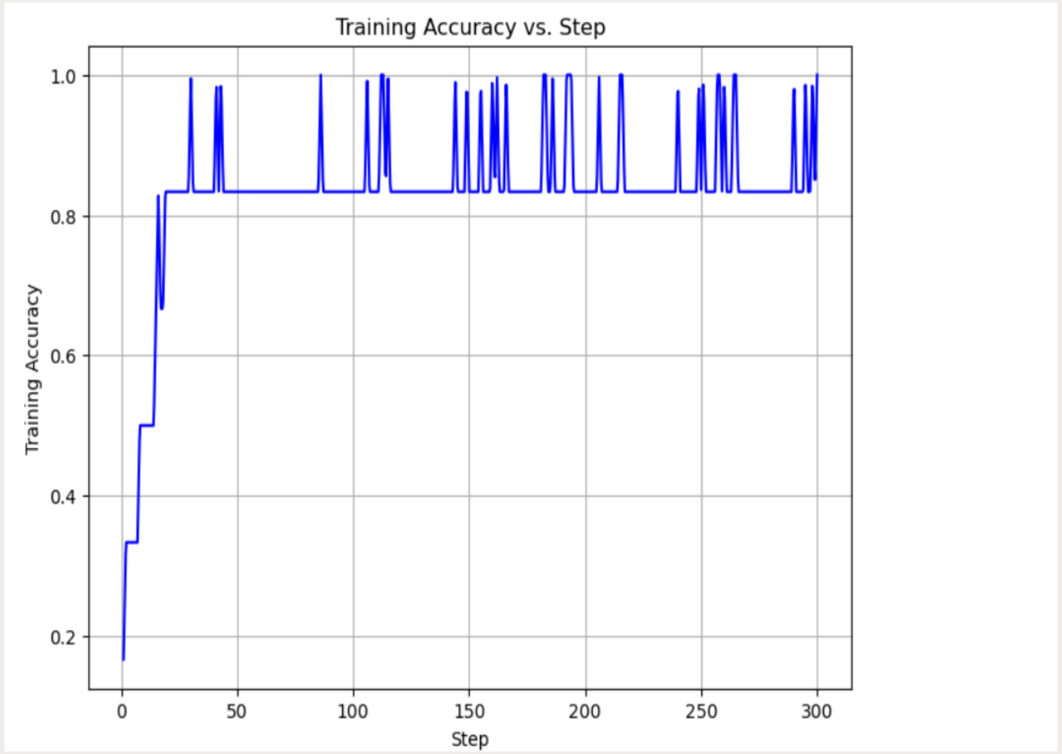
Afterwards, Inference was made by calling the model and testing it on new data. I made inference using 21 new entries and compared the actual price and the predicted price given To check its accuracy and derive other parameters like the RMSE, R^2 Score and MAE



```
Actual prices: ['$65.00', '$219.00', '$475.00', '$60.00', '$129.00', '$99.00', '$69.00', '$119.00', '$79.00', '$150.00', '$250.
00', '$250.00', '$250.00', '$75.00', '$250.00', '$110.00', '$80.00', '$70.00', '$165.00', '$70.00', '$50.00', '$40.00', '$125.0
0', '$245.00', '$110.00', '$150.00', '$165.00', '$150.00', '$150.00']
Predicted prices: ['$80.00', '$180.00', '$350.00', '$79.00', '$150.00', '$150.00', '$80.00', '$80.00', '$80.00', '$140.00', '$2
25.00', '$120.00', '$150.00', '$80.00', '$225.00', '$120.00', '$120.00', '$70.00', '$180.00', '$70.00', '$80.00', '$65.00', '$1
20.00', '$225.00', '$80.00', '$150.00', '$180.00', '$225.00', '$180.00']
```

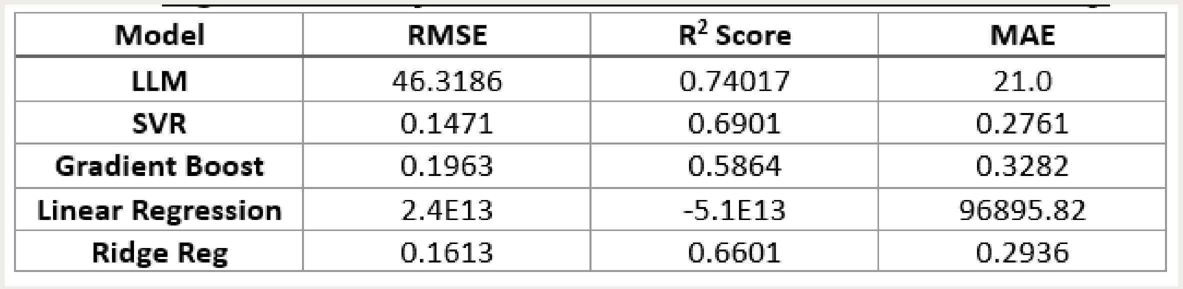
Training accuracy over each step

- The graph shows the training accuracy over each step and from it we can see that the models training accuracy looks promising as each step goes by, the accuracy tends to increase. This behaviour shows the capability of the LLM to discover underlying patterns provided a given set of features and adapt to it quickly.



Accuracy Comparison Of The LLM Against Other Models

- When the LLM is compared to other Models In terms of their metrics we can see that although it performs less in terms of RMSE AND MAE compared to the other models except the linear regressor model, it does extremely well in terms of its R squared score against other models. This underscores the LLM's ability to explain a larger proportion of the variance in the Airbnb price data, making it an attractive option, especially when interpretability is crucial.



Model	RMSE	R^2 Score	MAE
LLM	46.3186	0.74017	21.0
SVR	0.1471	0.6901	0.2761
Gradient Boost	0.1963	0.5864	0.3282
Linear Regression	2.4E13	-5.1E13	96895.82
Ridge Reg	0.1613	0.6601	0.2936

Questions

- Are traditional methods of price prediction going to be obsolete?
- Could the fine-tuning approach used in this project be applied to other domains beyond Airbnb price prediction?
- In what scenarios or contexts might traditional methods still be preferable?
- What impact could the adoption of LLM-based predictions have on the field of price prediction?
- Were there trade-offs or challenges in finding the right balance during fine-tuning?
- What considerations guided your decisions during the fine-tuning process, especially in terms of hyperparameters?
- What challenges or opportunities might arise when adapting this approach to different prediction tasks?

Conclusion

- The model produced satisfactory results with an exceptional coefficient score however, Adding more entries and more features into the training and validation set and testing out different hyperparameters would not only improve the models accuracy but make it more viable in price prediction.
- Neverthless, the model has demonstrated an interesting and decent amount of accuracy despite being trained with less than 1% of the entries provided in the dataset.
- This project shows the intriguing and promising impact Large Language models can have in the future of price prediction.