Create chatbot in python

Introduction:

- The real estate market is one of the most dynamic and lucrative sectors, with house

prices constantly fluctuating based on various factors such as location, size, amenities, and

economic conditions. Accurately predicting house prices is crucial for both buyers and

sellers, as it can help make informed decisions regarding buying, selling, or investing in

properties.

- Traditional linear regression models are often employed for house price prediction.

However, they may not capture complex relationships between predictors and the target

variable, leading to suboptimal predictions. In this project, we will explore advanced

regression techniques to enhance the accuracy and robustness of house price prediction

models.

- Briefly introduce the real estate market and the importance of accurate house price

prediction.

Highlight the limitations of traditional linear regression models in capturing complex

relationships.

- Emphasize the need for advanced regression techniques like Gradient Boosting and

XGBoost to enhance prediction accuracy.

Content for Project Phase 2 :

Consider exploring advanced regression techniques like Gradient Boosting or XGBoost for

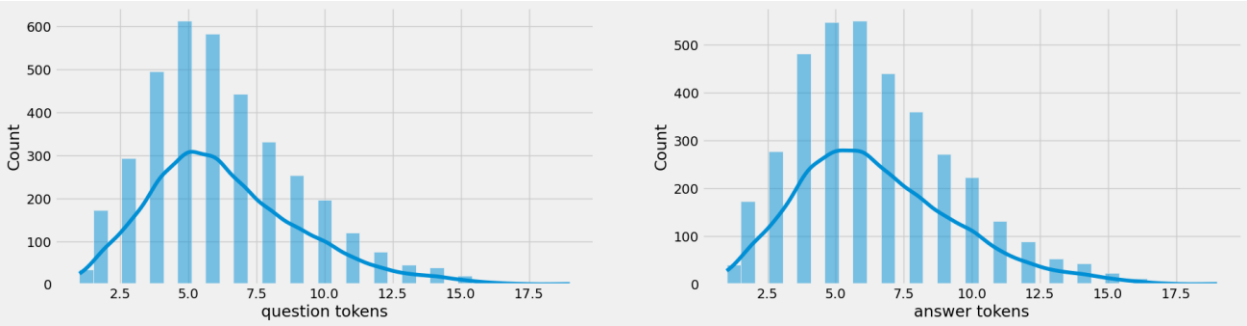
improved Prediction accuracy.

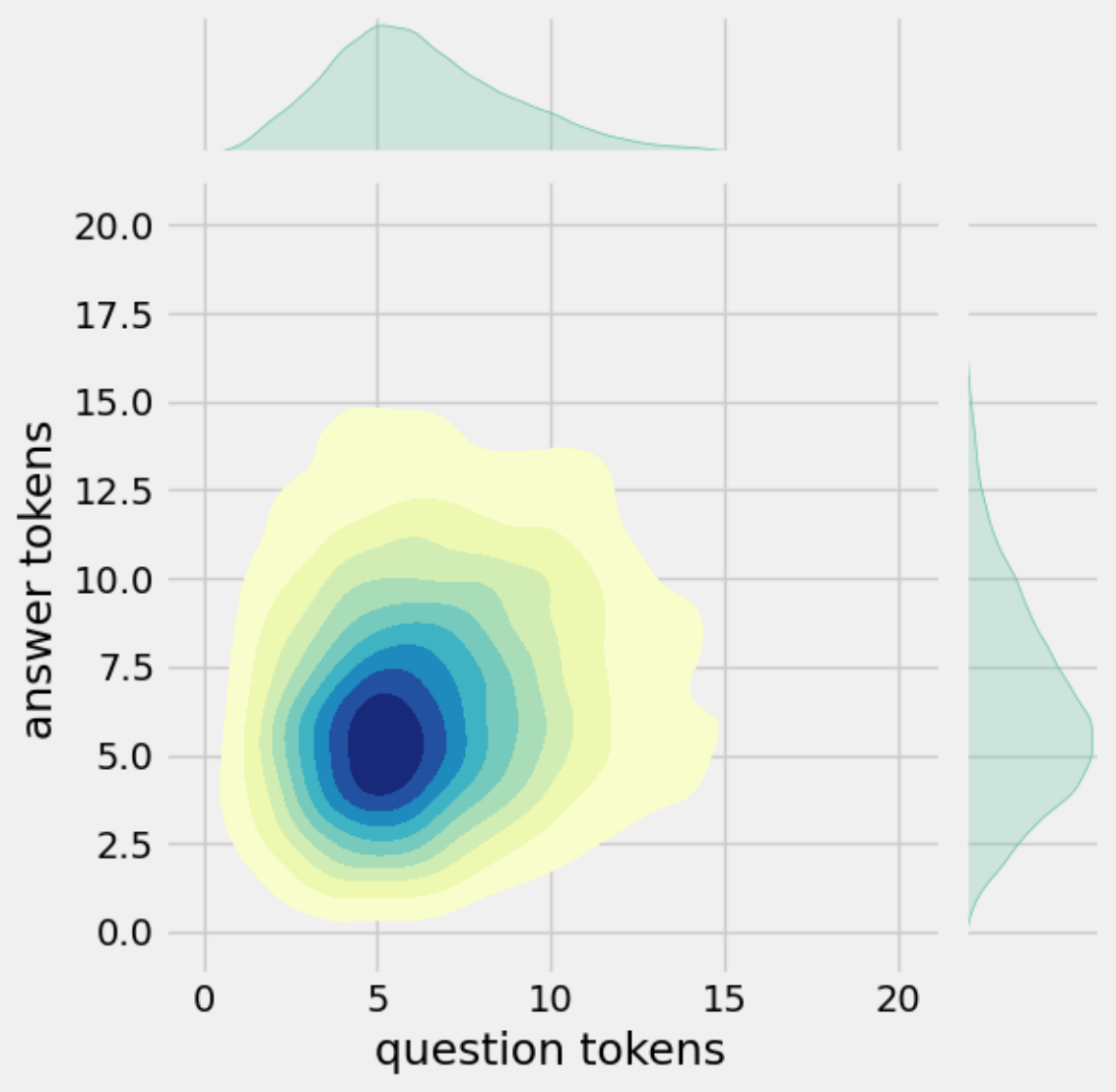
Data Source

A good data source for house price prediction using machine learning should be

Accurate, Complete, Covering the geographic area of interest, Accessible.

```
df['question tokens']=df['question'].apply(lambda x:len(x.split()))

df['answer tokens']=df['answer'].apply(lambda x:len(x.split()))

plt.style.use('fivethirtyeight')

fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))

sns.set_palette('Set2')

sns.histplot(x=df['question tokens'],data=df,kde=True,ax=ax[0])

sns.histplot(x=df['answer tokens'],data=df,kde=True,ax=ax[1])

sns.jointplot(x='question tokens',y='answer
tokens',data=df,kind='kde',fill=True,cmap='YlGnBu')

plt.show()
```





Data Collection and Preprocessing:

- ❖ Importing the dataset: Obtain a comprehensive dataset containing relevant features

such as square footage, number of bedrooms, location, amenities, etc.

- ❖ Data preprocessing: Clean the data by handling missing values, outliers, and

categorical variables. Standardize or normalize numerical features.

Exploratory Data Analysis (EDA):

- ❖ Visualize and analyze the dataset to gain insights into the relationships between

variables.

- ❖ Identify correlations and patterns that can inform feature selection and engineering.
- ❖ Present various data visualizations to gain insights into the dataset.
- ❖ Explore correlations between features and the target variable (house prices).
- ❖ Discuss any significant findings from the EDA phase that inform feature selection.

Feature Engineering:

- ❖ Create new features or transform existing ones to capture valuable information.
- ❖ Utilize domain knowledge to engineer features that may impact house prices, such as
  proximity to schools, transportation, or crime rates.
- ❖ Explain the process of creating new features or transforming existing ones.
- ❖ Showcase domain-specific feature engineering, such as proximity scores or composite

indicators.

- ❖ Emphasize the impact of engineered features on model performance.

Advanced Regression Techniques:

- ❖ Ridge Regression: Introduce L2 regularization to mitigate multicollinearity and

overfitting.

- ❖ Lasso Regression: Employ L1 regularization to perform feature selection and
  simplify the model.
- ❖ ElasticNet Regression: Combine both L1 and L2 regularization to benefit from their
  respective advantages.
- ❖ Random Forest Regression: Implement an ensemble technique to handle nonlinearity

and capture complex relationships in the data.

- ❖ Gradient Boosting Regressors (e.g., XGBoost, LightGBM): Utilize gradient
- ❖ boosting algorithms for improved accuracy.

Model Evaluation and Selection:

Split the dataset into training and testing sets.

- ❖ Evaluate models using appropriate metrics (e.g., Mean Absolute Error, Mean Squared

Error, R-squared) to assess their performance.

- ❖ Use cross-validation techniques to tune hyperparameters and ensure model stability.
- ❖ Compare the results with traditional linear regression models to highlight

improvements.

- ❖ Select the best-performing model for further analysis.

Model Interpretability:

- ❖ Explain how to interpret feature importance from Gradient Boosting and XGBoost

models.

- ❖ Discuss the insights gained from feature importance analysis and their relevance to
  house price prediction.
- ❖ Interpret feature importance from ensemble models like Random Forest and Gradient
- ❖ Boosting to understand the factors influencing house prices.
- ❖ **Deployment and Prediction:**
- ❖ Deploy the chosen regression model to predict house prices.
- ❖ Develop a user-friendly interface for users to input property features and receive price
- ❖ predictions.

Program:

Creat chatbot

```
print(f"After preprocessing: {' '.join(df[df['encoder input tokens'].max()==df['encoder input tokens']]['encoder_inputs'].values.tolist())}")

print(f"Max encoder input length: {df['encoder input tokens'].max()}")

print(f"Max decoder input length: {df['decoder input tokens'].max()}")

print(f"Max decoder target length: {df['decoder target tokens'].max()}")


df.drop(columns=['question','answer','encoder input tokens','decoder input tokens','decoder target tokens'],axis=1,inplace=True)

params={
    "vocab_size":2500,
    "max_sequence_length":30,
    "learning_rate":0.008,
    "batch_size":149,
    "lstm_cells":256,
    "embedding_dim":256,
    "buffer_size":10000
}

learning_rate=params['learning_rate']

batch_size=params['batch_size']

embedding_dim=params['embedding_dim']

lstm_cells=params['lstm_cells']

vocab_size=params['vocab_size']

buffer_size=params['buffer_size']

max_sequence_length=params['max_sequence_length']

df.head(10)
```

After preprocessing: for example , if your birth date is january 1 2 , 1 9 8 7 , write 0 1 / 1 2 / 8 7 .

Max encoder input length: 27

Max decoder input length: 29

Max decoder target length: 28

Out[2]:

	encoder_inputs	decoder_targets	decoder_inputs
0	hi , how are you doing ?	i ' m fine . how about yourself ? <end>	<start> i ' m fine . how about yourself ? <end>
1	i ' m fine . how about yourself ?	i ' m pretty good . thanks for asking . <end>	<start> i ' m pretty good . thanks for asking...
2	i ' m pretty good . thanks for asking .	no problem . so how have you been ? <end>	<start> no problem . so how have you been ? ...
3	no problem . so how have you been ?	i ' ve been great . what about you ? <end>	<start> i ' ve been great . what about you ? ...
4	i ' ve been great . what about you ?	i ' ve been good . i ' m in school right now ...	<start> i ' ve been good . i ' m in school ri...
5	i ' ve been good . i ' m in school right now .	what school do you go to ? <end>	<start> what school do you go to ? <end>
6	what school do you go to ?	i go to pcc . <end>	<start> i go to pcc . <end>
7	i go to pcc .	do you like it there ? <end>	<start> do you like it there ? <end>
8	do you like it there ?	it ' s okay . it ' s a really big campus . <...	<start> it ' s okay . it ' s a really big cam...
9	it ' s okay . it ' s a really big campus .	good luck with school . <end>	<start> good luck with school . <end>

Build Models

Build Encoder

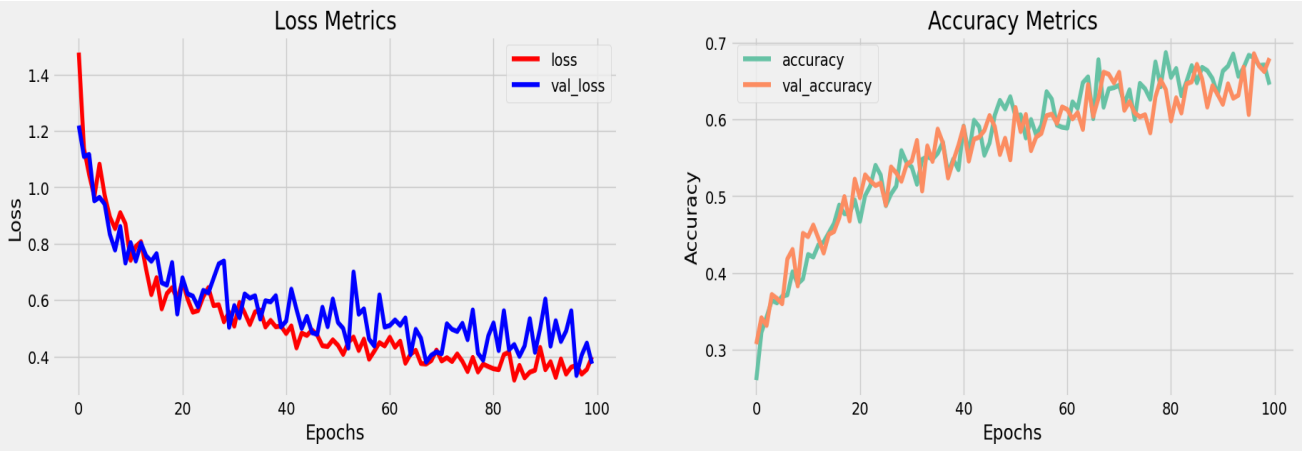
```
class Encoder(tf.keras.models.Model):
    def __init__(self,units,embedding_dim,vocab_size,*args,**kwargs) -> None:
        super().__init__(*args,**kwargs)
        self.units=units
        self.vocab_size=vocab_size
        self.embedding_dim=embedding_dim
```

```
        self.embedding=Embedding(
            vocab_size,
            embedding_dim,
            name='encoder_embedding',
            mask_zero=True,
            embeddings_initializer=tf.keras.initializers.GlorotNormal()
        )
        self.normalize=LayerNormalization()
        self.lstm=LSTM(
            units,
            dropout=.4,
            return_state=True,
            return_sequences=True,
            name='encoder_lstm',
            kernel_initializer=tf.keras.initializers.GlorotNormal()
        )

    def call(self,encoder_inputs):
        self.inputs=encoder_inputs
        x=self.embedding(encoder_inputs)
        x=self.normalize(x)
        x=Dropout(.4)(x)
        encoder_outputs,encoder_state_h,encoder_state_c=self.lstm(x)
        self.outputs=[encoder_state_h,encoder_state_c]
        return encoder_state_h,encoder_state_c
```

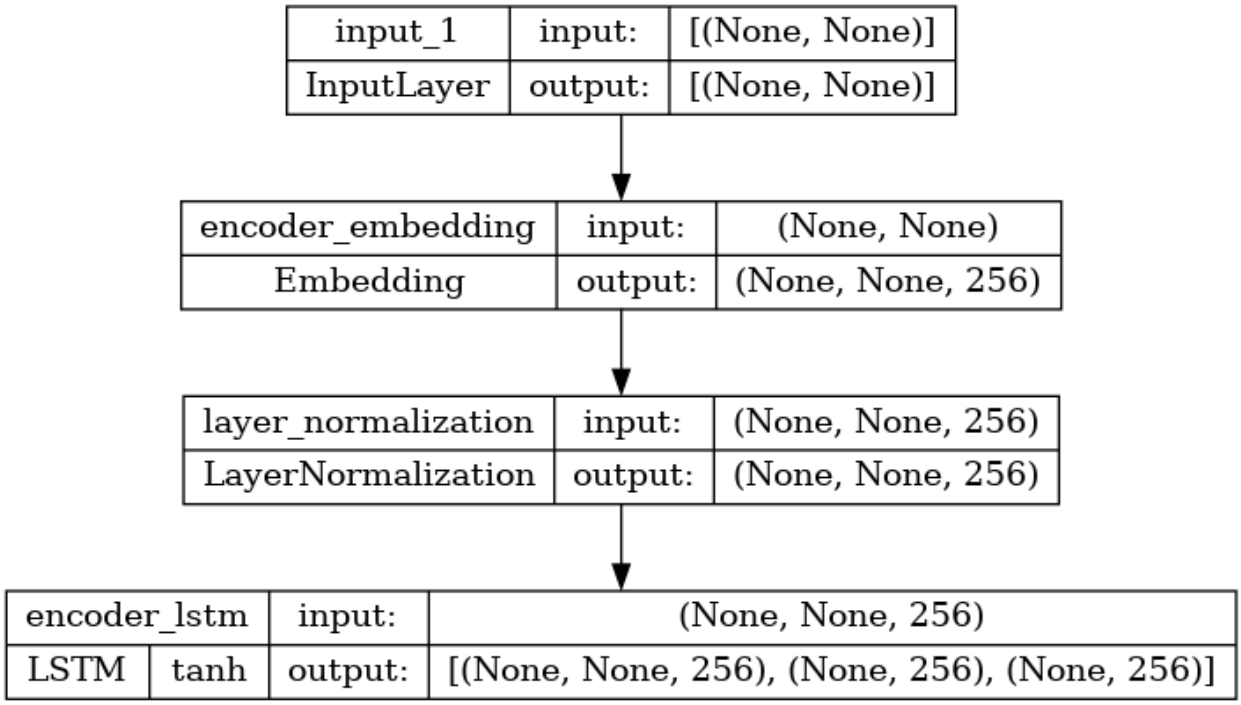
```
encoder=Encoder(lstm_cells,embedding_dim,vocab_size,name='encoder')
encoder.call(_[0])
```

Visualize Metrics

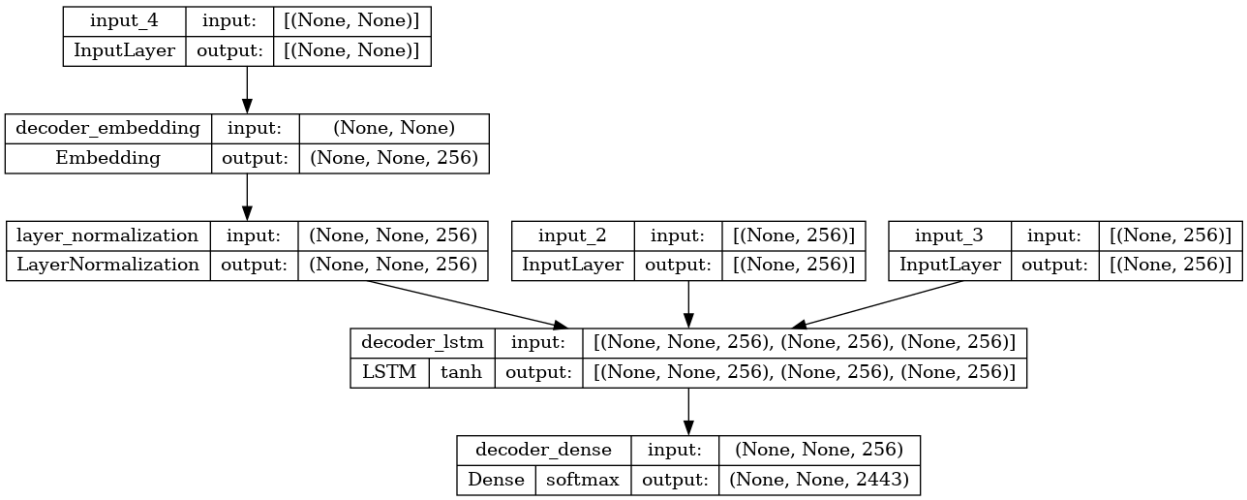
```
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))
ax[0].plot(history.history['loss'],label='loss',c='red')
ax[0].plot(history.history['val_loss'],label='val_loss',c = 'blue')
ax[0].set_xlabel('Epochs')
ax[1].set_xlabel('Epochs')
ax[0].set_ylabel('Loss')
ax[1].set_ylabel('Accuracy')
ax[0].set_title('Loss Metrics')
ax[1].set_title('Accuracy Metrics')
ax[1].plot(history.history['accuracy'],label='accuracy')
ax[1].plot(history.history['val_accuracy'],label='val_accuracy')
ax[0].legend()
ax[1].legend()
plt.show()
```



tf.keras.utils.plot_model(chatbot.encoder,to_file='encoder.png',show_shapes=True, show_layer_activations=True)



tf.keras.utils.plot_model(chatbot.decoder,to_file='decoder.png',show_shapes=True, show_layer_activations=True)



Time to Chat

```
def print_conversation(texts):
    for text in texts:
        print(f'You: {text}')
        print(f'Bot: {chatbot(text)}')
        print('==========================')
print_conversation([
    'hi',
    'do yo know me?',
    'what is your name?',
    'you are bot?',
    'hi, how are you doing?',
    "i'm pretty good. thanks for asking.",
    "Don't ever be in a hurry",
    "'I'm gonna put some dirt in your eye '",
    "'You're trash '",
    "'I've read all your research on nano-technology '",
    "'You want forgiveness? Get religion'",
    "'While you're using the bathroom, i'll order some food.'",
    "'Wow! that's terrible.'",
    "'We'll be here forever.'",
    "'I need something that's reliable.'",
    "'A speeding car ran a red light, killing the girl.'",
    "'Tomorrow we'll have rice and fish for lunch.'",
    "'I like this restaurant because they give you free bread.'"
```

])

Project Conclusion:

- In the Phase 2 conclusion, we will summarize the key findings and insights from the

advanced regression techniques. We will reiterate the impact of these techniques on

improving the accuracy and robustness of house price predictions.

- **Future Work:** We will discuss potential avenues for future work, such as incorporating

additional data sources (e.g., real-time economic indicators), exploring deep learning models

for prediction, or expanding the project into a web application with more features and

interactivity.