VERSION 1 Explanation

1. **Data Collection:** Scrapes property listing data from specified URLs using requests and BeautifulSoup.
2. **Data Preprocessing:** Cleans the raw data (e.g., removing duplicates, converting price strings to numbers).
3. **Exploratory Data Analysis:** Visualizes data distributions and trends with Matplotlib and Seaborn.
4. **Feature Engineering:** Generates new features such as a proximity score.
5. **Correlation Analysis:** Computes and visualizes a correlation heatmap.
6. **Model Selection:** Compares several regression models using Mean Squared Error.
7. **Model Training:** Fits the chosen model on training data.
8. **Model Evaluation:** Computes MAE, MSE, and R^2 on test data.
9. **Deployment:** Provides an example Flask API endpoint for predictions.
10. **Validation and Iteration:** Demonstrates how new data can be used for validation and subsequent model improvements.

Dev Parekh Version1.py

VERSION 2 Explanation

Python-Based Implementation Methodology (15 Steps)

---

1. Environment Setup & Legal Compliance

```
# Install required libraries
!pip install pandas numpy matplotlib seaborn scikit-learn beautifulsoup4 scrapy selenium flask streamlit
```

- Verify scraping permissions via platforms' robots.txt
- Obtain API keys if available (e.g., Google Maps for geocoding)

---

2. Data Collection via Web Scraping

```
from bs4 import BeautifulSoup
import scrapy
from selenium import webdriver

# Example: Scrape MagicBricks listings
driver = webdriver.Chrome()
driver.get("https://www.magicbricks.com")
soup = BeautifulSoup(driver.page_source, "html.parser")
# Extract price, location, amenities, etc.
```

---

3. Data Storage & Organization

- Save scraped data to structured format:

```
import pandas as pd
```

```python
df = pd.DataFrame({
    "price": [...],
    "location": [...],
    "sq_ft": [...],
    "amenities": [...]
})
df.to_csv("real_estate_data.csv", index=False)
```

---

4. Geospatial Feature Engineering

```python
from geopy.geocoders import Nominatim

# Convert addresses to coordinates
geolocator = Nominatim(user_agent="real_estate_app")
df["coordinates"] = df["location"].apply(lambda x: geolocator.geocode(x).point)
```

---

5. Proximity Scoring

```python
from math import radians, sin, cos, sqrt, atan2

def calculate_distance(coord1, coord2):
    # Haversine formula implementation
    return distance_km

# Score proximity to schools/hospitals
df["school_distance"] = df["coordinates"].apply(lambda x: calculate_distance(x, school_coord))
```

---

6. Data Preprocessing

```
# Handle missing values
df = df.dropna()

# Normalize numerical features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[["sq_ft", "school_distance"]] = scaler.fit_transform(df[["sq_ft", "school_distance"]])

# Encode categorical variables
df = pd.get_dummies(df, columns=["property_type"])
```

---

7. Exploratory Data Analysis (EDA)

```
import seaborn as sns

# Price distribution
sns.histplot(df["price"], kde=True)

# Correlation heatmap
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True)
```

---

8. Feature Selection

```
from sklearn.feature_selection import SelectKBest, f_regression

selector = SelectKBest(score_func=f_regression, k=10)
X_selected = selector.fit_transform(X_train, y_train)
```

---

9. Model Development

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = GradientBoostingRegressor(n_estimators=200, learning_rate=0.05)
model.fit(X_train, y_train)
```

---

10. Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    "n_estimators": [100, 200],
    "max_depth": [3, 5]
}
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

---

11. Model Evaluation

```
from sklearn.metrics import mean_absolute_error, r2_score

y_pred = model.predict(X_test)
print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
print(f"R²: {r2_score(y_test, y_pred)}")
```

---

12. Explainability Analysis

```
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test)
```

---

13. Deployment with Streamlit

```
# app.py
import streamlit as st

def predict_price(inputs):
   return model.predict(inputs)

st.title("Real Estate Price Predictor")
st.slider("Square Feet", 500, 5000)
if st.button("Predict"):
   st.write(f"Estimated Price: ${predict_price(...):.2f}")
```

---

14. Validation & Iteration

- Collect user feedback via the deployed app
- Retrain model monthly with new data

```
model.partial_fit(new_X, new_y)  # Online learning
```

---

15. Documentation & Reporting

- Generate model cards with torch.utils.tensorboard
- Export results to LaTeX/PDF:

```
!pdflatex final_report.tex
```

Key Libraries Used:

- Data Handling: Pandas, NumPy
- Visualization: Matplotlib, Seaborn, SHAP
- ML: Scikit-learn, XGBoost
- Deployment: Streamlit

VERSION 3 Explanation

Dev Parekh Version3.py

Python-Based Implementation Methodology for Real Estate Pricing Model

This methodology outlines a 15-step process to develop a data-driven real estate pricing model using Python, as proposed in the research report "A Data-Driven Approach to Real Estate Pricing." Each step leverages specific Python libraries and techniques to achieve the research objectives.

This 15-step methodology provides a comprehensive Python-based framework to develop a fair and transparent real estate pricing model. By systematically addressing data collection, analysis, modeling, and deployment, it aligns with the research goal of creating a data-driven valuation system.

1. Data Collection

- **Objective**: Gather real estate data from online platforms.
- **Description**: ScrapE property listings from websites like MagicBricks and Housing.com, focusing on features such as ***location, price, size, proximity to services (schools, hospitals, transport), infrastructure quality, amenities, and legal factors***.
- **Tools**: requests, BeautifulSoup, Scrapy, Selenium, pandas.

2. Data Preprocessing

- **Objective**: Clean and prepare the data for analysis.
- **Description**: Remove duplicates, handle missing values, and standardize data formats. Convert categorical variables (e.g., location) into numerical formats using one-hot encoding and normalize numerical features (e.g., price, size).
- **Tools**: pandas, NumPy.

3. Exploratory Data Analysis (EDA)

- **Objective**: Uncover patterns and relationships in the data.
- **Description**: Visualize distributions of key variables (***e.g., price***) and relationships (e***.g., price vs. proximity to services***) using plots like histograms, scatter plots, and heatmaps. Analyze categorical impacts ***(e.g., property type).***
- **Tools**: Matplotlib, Seaborn, pandas.

4. Feature Engineering

- **Objective**: Enhance the dataset with derived features.
- **Description**: Create features like distance scores to essential services, amenity indexes (***e.g., count of gyms, parks),*** and infrastructure quality metrics ***(e.g., road condition scores).***

- **Tools**: pandas, NumPy, geopy (for geospatial calculations).

5. Correlation Analysis

- **Objective**: Identify key factors influencing prices.
- **Description**: Compute correlation coefficients ***(e.g., Pearson)*** between features and price. Use statistical tests to validate significance.
- **Tools**: pandas, SciPy, statsmodels.

6. Model Selection

- **Objective**: Choose suitable machine learning models.
- **Description**: Evaluate regression models like ***Linear Regression, Random Forest, and Gradient Boosting*** based on data characteristics ***(e.g., linearity, feature interactions).***
- **Tools**: Scikit-learn.

7. Model Training

- **Objective**: Train models on the prepared dataset.
- **Description**: Split data into training (80%) and testing (20%) sets. Tune hyperparameters using ***Grid Search or Random Search*** and train models.
- **Tools**: Scikit-learn, pandas.

8. Model Evaluation

- **Objective**: Assess model performance.
- **Description**: Calculate metrics like ***Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.*** Compare models and analyze residuals for improvements.
- **Tools**: Scikit-learn, Matplotlib, Seaborn.

9. Model Deployment

- **Objective**: Deploy the model for practical use.
- **Description**: Use ***Streamlit*** to create a web interface where users can input property details and receive price predictions.
- **Tools**: Streamlit, pickle (for model saving).

10. Validation and Iteration

- **Objective**: Refine the model over time.
- **Description**: Gather user feedback and new data to retrain the model, improving accuracy and relevance. Adjust features and models as needed.
- **Tools**: pandas, Scikit-learn.

11. Handling Qualitative Factors

- **Objective**: Incorporate subjective elements like neighbourhood perception.

- **Description**: Apply sentiment analysis to property reviews and integrate external data ***(e.g., crime rates)*** to quantify qualitative aspects.
- **Tools**: NLTK, TextBlob, pandas.

12. Legal and Governance Integration

- **Objective**: Include legal and governance impacts.
- **Description**: Collect data on zoning policies and property taxes, engineering features to reflect their influence on value.
- **Tools**: pandas, NumPy.

13. Scalability and Performance Optimization

- **Objective**: Ensure efficiency with large datasets.
- **Description**: Optimize processing with parallel computing or batch methods to handle scalability.
- **Tools**: Dask, NumPy, pandas.

14. Documentation and Reporting

- **Objective**: Record the process for transparency.
- **Description**: Document data sources, preprocessing, and model details. Generate reports with visualizations to summarize findings.
- **Tools**: Jupyter Notebook, Matplotlib, Seaborn.

15. Ethical Considerations and Bias Mitigation

- **Objective**: Ensure fairness in predictions.
- **Description**: Check for biases (e.g., location-based) and apply mitigation techniques like reweighting or fairness constraints.
- **Tools**: Scikit-learn, AIF360.