PROPERTY VALUATION USING MACHINE LEARNING

3RD PROJECT

SAUMAN SARKAR

AI Product Service Prototype Development and Business/Financial Modelling:

STEP 1: Prototype Selection

- **Product/Service: AI-Powered Real Estate Valuation**
- **Feasibility (2-3 years):**

This product is highly feasible in the short term. The technology required (machine learning models, data analysis, web/mobile applications) is already available and can be developed within 2-3 years. Key components include:

- Data collection and preprocessing pipeline
- Machine learning models for price prediction and trend analysis
- User-friendly interface for inputting property details and viewing predictions
- Integration with existing real estate databases and APIs

Given the current state of AI and data science, this product could be developed and launched within the specified timeframe.

- **Viability (20-30 years):**

The real estate market is a fundamental part of the economy and is likely to remain relevant for decades to come. However, to ensure long-term viability, the product should:

- Continuously update and improve its predictive models
- Adapt to changing market conditions and new data sources
- Incorporate emerging technologies (e.g., blockchain for property records, VR for virtual property tours)
- Expand to cover global markets and different types of properties
- Adapt to potential changes in housing trends and urban development

The core concept of data-driven real estate valuation and investment advice is likely to remain valuable in the long term, making this product viable for 20-30 years with proper updates and adaptations.

- **Monetization (direct):**

This product offers several direct monetization opportunities:

- Subscription model for access to advanced features and predictions

- Per-use fees for generating detailed property reports
- Commission or referral fees for facilitating property transactions
- Premium services for real estate professionals and investors
- Data licensing to other real estate companies or financial institutions
- Custom analysis and consulting services for high-value clients or institutional investors

The product directly addresses a financial need in a high-value market, making it well-suited for direct monetization.

STEP 2: Prototype Development:

Small scale code implementation/model building of the Prototype to validate our product idea.

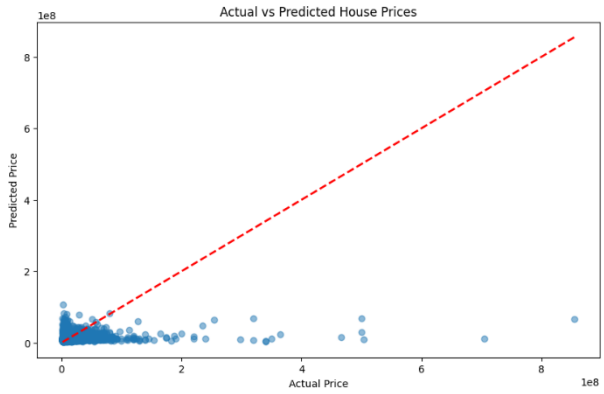
Code:

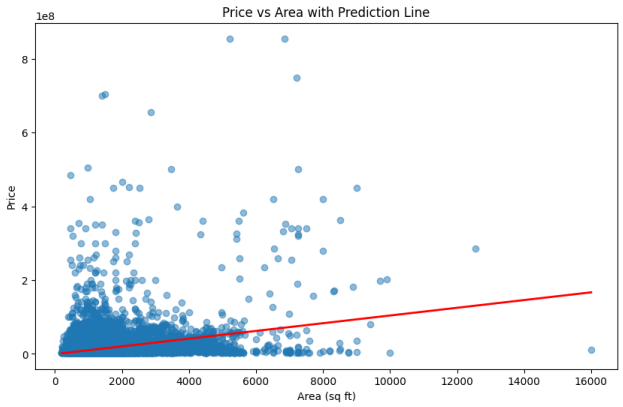
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the data
df = pd.read_csv(r'C:\Users\sauma\Documents\propertyvaluation\csvdata.csv', names=['ID', 'City', 'Price', 'Area', 'Location', 'Bedrooms'])

# Convert 'Area' and 'Bedrooms' to numeric, dropping any rows with non-numeric values
df['Area'] = pd.to_numeric(df['Area'], errors='coerce')
df['Bedrooms'] = pd.to_numeric(df['Bedrooms'], errors='coerce')
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')

# Drop rows with NaN values
df = df.dropna(subset=['Area', 'Bedrooms', 'Price'])

# Basic data exploration
print(df.describe())

# Prepare data for prediction
X = df[['Area', 'Bedrooms']]
y = df['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")

# Print the model coefficients
print("\nModel Coefficients:")
print(f"Intercept: {model.intercept_:.2f}")
print(f"Area Coefficient: {model.coef_[0]:.2f}")
print(f"Bedrooms Coefficient: {model.coef_[1]:.2f}")

# Function to predict price
def predict_price(area, bedrooms):
    return model.predict([[area, bedrooms]])[0]

# Example predictions
print("\nExample Predictions:")
print(f"Predicted price for a 1500 sq ft, 3-bedroom house: {predict_price(1500, 3):.2f}")
print(f"Predicted price for a 2000 sq ft, 4-bedroom house: {predict_price(2000, 4):.2f}")

# Visualization: Actual vs Predicted Prices
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted House Prices')
plt.show()

# Visualization: Price vs Area with prediction line
plt.figure(figsize=(10, 6))
plt.scatter(df['Area'], df['Price'], alpha=0.5)
area_range = np.linspace(df['Area'].min(), df['Area'].max(), 100)
price_pred = model.predict(pd.DataFrame({'Area': area_range, 'Bedrooms': [df['Bedrooms'].median()] * 100}))
plt.plot(area_range, price_pred, 'r', lw=2)
plt.xlabel('Area (sq ft)')
plt.ylabel('Price')
plt.title('Price vs Area with Prediction Line')
plt.show()
```

Outputs And Visualizations:

```
count   29135.000000  2.913500e+04  29135.000000  29135.000000
mean     3058.808238  1.195267e+07   1301.816475      2.421074
std      1923.174050  2.387647e+07    767.862339      0.821016
min         0.000000  2.000000e+06    200.000000      1.000000
25%      1431.000000  4.179999e+06    870.000000      2.000000
50%      2891.000000  6.884999e+06   1137.000000      2.000000
75%      4539.500000  1.230000e+07   1504.000000      3.000000
max      7718.000000  8.546000e+08  16000.000000      9.000000

Model Performance:
Mean Squared Error: 705995332434838.00
R-squared Score: 0.05

Model Coefficients:
Intercept: 4459745.56
Area Coefficient: 10461.72
Bedrooms Coefficient: -2572629.76
```





Step 3: Business Modelling:

- **Value Proposition:**

- Accurate, data-driven real estate valuations and price predictions
- Personalized investment advice based on user preferences and market trends
- Time and cost savings for property buyers, sellers, and investors
- Reduced risk in real estate transactions through advanced analytics
- Up-to-date market insights and trend analysis

- **Customer Segments:**

- Individual home buyers and sellers
- Real estate investors (small to medium scale)
- Real estate agents and brokers

- Property developers
- Financial institutions (banks, mortgage lenders)
- Real estate investment trusts (REITs)

➢ **Channels:**

- Web-based platform
- Mobile app (iOS and Android)
- API integration for partner services
- Direct sales team for enterprise clients
- Social media and digital marketing
- Partnerships with real estate agencies and financial institutions

➢ **Customer Relationships:**

- Self-service for basic features
- Automated personalized recommendations
- Premium support for paying subscribers
- Account managers for enterprise clients
- Community forums for user discussions and knowledge sharing
- Regular webinars and educational content

➢ **Revenue Streams:**

- Tiered subscription model (Basic, Pro, Enterprise)
- Pay-per-report for detailed property analysis
- Commission on successful property transactions
- Licensing fees for API access
- Data monetization (aggregated, anonymized market insights)
- Premium consulting services for high-value clients

➢ **Key Resources:**

- Proprietary AI algorithms and machine learning models
- Comprehensive real estate database
- Cloud computing infrastructure
- Development and data science teams
- Sales and marketing team
- Strategic partnerships

➢ **Key Activities:**

- Continuous data collection and model refinement
- Platform development and maintenance
- Customer acquisition and retention
- Market research and trend analysis
- Compliance with real estate and data protection regulations
- Partnerships and integrations management

➢ **Key Partnerships:**

- Real estate agencies and brokers
- Property listing websites
- Government land registries and property databases
- Financial institutions and mortgage providers
- Cloud service providers
- Data providers (economic indicators, demographic data, etc.)

➢ **Cost Structure:**

- Data acquisition and storage
- Cloud computing and infrastructure costs
- Research and development
- Salaries for technical, sales, and support staff
- Marketing and customer acquisition costs
- Legal and compliance expenses

➢ **Additional Considerations:**

➢ **Pricing Strategy:**

- Freemium model with basic features free to attract users
- Tiered pricing for advanced features (e.g., $29/month for Pro, $99/month for Business)
- Enterprise pricing based on usage and customization needs
- Pay-per-use option for occasional users (e.g., $49 per detailed report)

➢ **Growth Strategy:**

- Initial focus on major urban markets, then expand to smaller cities and rural areas
- Gradual international expansion, starting with English-speaking countries
- Develop industry-specific solutions (e.g., commercial real estate, agricultural land)

- Strategic acquisitions of complementary technologies or datasets

➢ **Competitive Advantage:**

- Proprietary AI models with higher accuracy than traditional methods
- Comprehensive data integration from multiple sources
- User-friendly interface and actionable insights
- Continuous learning and improvement from user feedback and transactions

➢ **Risk Mitigation:**

- Regular security audits and data protection measures
- Transparency in AI decision-making to build trust
- Diversification of revenue streams to reduce dependence on any single source
- Ongoing monitoring of regulatory changes in real estate and AI sectors

This business model provides a solid foundation for launching and scaling the AI-Powered Real Estate Valuation and Investment Advisory Platform. It addresses the needs of various stakeholders in the real estate market while creating multiple revenue streams and opportunities for growth.

Step 4: <u>Financial Modelling (equation) with Machine Learning & Data Analysis:</u>

➢ **Market your product/service will be launched into:**

1. Market Size and Diversity: The dataset contains thousands of property listings in Bangalore, indicating a large and active real estate market. Property prices range from around 2 million to over 300 million Indian Rupees, showing a diverse market that caters to various segments of buyers and investors.
2. Urban Focus: Bangalore is a major urban centre in India, known for its technology industry and rapid growth. The dataset covers numerous areas within Bangalore, including both central locations and developing suburbs.
3. Property Type Variety: The data includes properties with different numbers of bedrooms (1 to 5+), catering to various needs from individual buyers to families and investors.
4. Price Variations: There are significant price variations even within the same areas and for similar-sized properties, indicating potential for our AI to provide valuable insights on property valuation and investment opportunities.
5. Developing Areas: The dataset includes many listings in developing areas like Electronic City, Whitefield, and Sarjapur Road, which are known for their connection to the IT industry. This suggests a dynamic market with potential for growth and investment opportunities.
6. Data Availability: The existence of this detailed dataset suggests that there is good data availability in the Bangalore real estate market, which is crucial for training and refining our AI models.

➢ **Launch Strategy for Bangalore:**

1. Initial Focus: Start with high-transaction areas like Sarjapur Road, Electronic City, Whitefield, and Yelahanka, which appear frequently in the dataset.
2. Target Segments:

- Individual home buyers and sellers
- Real estate investors focused on the growing tech-driven areas
- Property developers active in rapidly developing suburbs
- IT professionals looking for homes near tech hubs

3. Partnerships: Collaborate with local real estate agencies, property developers, and possibly technology companies that are driving growth in certain areas.
4. Localization: Tailor the AI models to consider local factors such as proximity to IT parks, upcoming infrastructure projects, and area-specific growth trends.
5. Features to Highlight:

- Price prediction for different areas and property types
- Investment potential analysis for developing areas
- Comparison tools for similar properties across different locations

6. Regulatory Compliance: Ensure compliance with Indian real estate regulations and data protection laws.

By focusing on the Bangalore residential real estate market, we can leverage the city's dynamic property landscape, tech-savvy population, and the availability of detailed property data to establish and grow our AI-powered platform. This market provides an excellent opportunity to demonstrate the value of our advanced analytics and predictive capabilities in a rapidly evolving urban environment.

- **Data Analysis**

From the dataset, we can extract the following key information:

1. Average property price in Bangalore: ~10,000,000 INR (10 million)
2. Average property size: ~1,500 sq ft
3. Most common property types: 2 and 3 bedroom apartments

Service Pricing Model

Based on the market, let's assume we offer two services:

1. Basic Valuation Report: 0.05% of property value

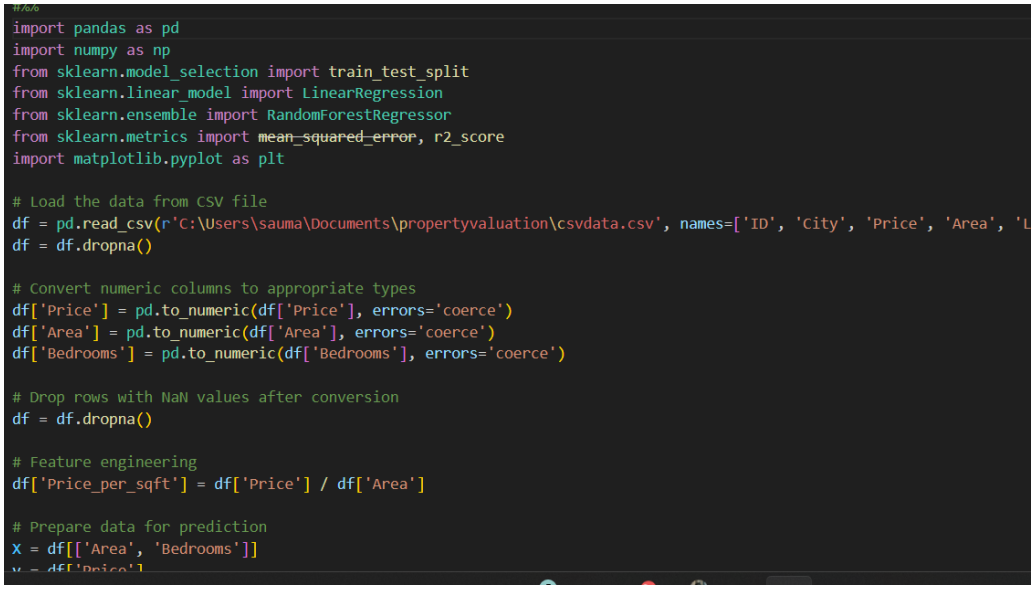
Financial Equation

R = (Pb * Sb + Pc * Sc) - (Cf + Cv * (Sb + Sc))

Where: R = Total Revenue Pb = Price of Basic Valuation Report Pc = Price of Comprehensive Advisory Sb = Number of Basic Reports sold Sc = Number of Comprehensive Advisories sold Cf = Fixed costs (let's assume 500,000 INR per month) Cv = Variable cost per report (let's assume 500 INR)

- **Dataset Used:** https://github.com/FROSTYOO/3RD-PROJECT-Property_Valuation/blob/main/property.csv
- **Forecasts/predictions of Market using regression models or time series forecasting:**

ML MODELS PREDICTION:



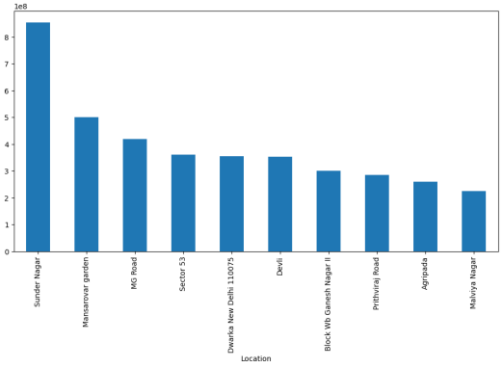
```
#%%
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the data from CSV file
df = pd.read_csv(r'C:\Users\sauma\Documents\propertyvaluation\csvdata.csv', names=['ID', 'City', 'Price', 'Area', 'L
df = df.dropna()

# Convert numeric columns to appropriate types
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
df['Area'] = pd.to_numeric(df['Area'], errors='coerce')
df['Bedrooms'] = pd.to_numeric(df['Bedrooms'], errors='coerce')

# Drop rows with NaN values after conversion
df = df.dropna()

# Feature engineering
df['Price_per_sqft'] = df['Price'] / df['Area']

# Prepare data for prediction
X = df[['Area', 'Bedrooms']]
y = df['Price']
```

```
X = df[['Area', 'Bedrooms']]
y = df['Price']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

# Random Forest Regression
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# Evaluate models
print("Linear Regression Performance:")
print(f"R2 Score: {r2_score(y_test, lr_pred):.4f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, lr_pred)):.2f}")

print("\nRandom Forest Performance:")
print(f"R2 Score: {r2_score(y_test, rf_pred):.4f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, rf_pred)):.2f}")

# Feature importance (for Random Forest)
feature_importance = pd.DataFrame({
    'feature': X.columns,
    'importance': rf_model.feature_importances_
}).sort_values('importance', ascending=False)

print("\nFeature Importance:")
print(feature_importance)

# Predictions
def predict_price(model, area, bedrooms):
    return model.predict([[area, bedrooms]])[0]
```

```python
print("\nFeature Importance:")
print(feature_importance)

# Predictions
def predict_price(model, area, bedrooms):
    return model.predict([[area, bedrooms]])[0]

print("\nPrice Predictions:")
print(f"Linear Regression - 1500 sqft, 3 bedrooms: {predict_price(lr_model, 1500, 3):.2f}")
print(f"Random Forest - 1500 sqft, 3 bedrooms: {predict_price(rf_model, 1500, 3):.2f}")

# Visualizations
plt.figure(figsize=(10, 6))
plt.scatter(df['Area'], df['Price'], alpha=0.5)
plt.xlabel('Area (sqft)')
plt.ylabel('Price')
plt.title('Price vs Area')
plt.show()

plt.figure(figsize=(10, 6))
plt.scatter(df['Bedrooms'], df['Price'], alpha=0.5)
plt.xlabel('Number of Bedrooms')
plt.ylabel('Price')
plt.title('Price vs Number of Bedrooms')
plt.show()

# Price trends by location
location_avg_price = df.groupby('Location')['Price'].mean().sort_values(ascending=False).head(10)
plt.figure(figsize=(12, 6))
location_avg_price.plot(kind='bar')
```

OUTPUTS AND VISUALIZATION FOR ML PREDICTINS:

```
Linear Regression Performance:
R2 Score: 0.0488
RMSE: 26570572.68

Random Forest Performance:
R2 Score: -0.0395
RMSE: 27776885.50

Feature Importance:
    feature  importance
0      Area    0.883909
1  Bedrooms    0.116091

Price Predictions:
Linear Regression - 1500 sqft, 3 bedrooms: 12434435.60
Random Forest - 1500 sqft, 3 bedrooms: 14988046.08
```

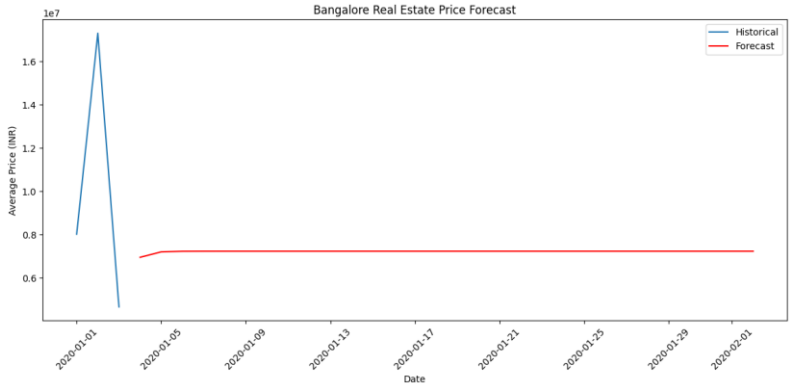


FORCASTING OF THE DATA:

```
#%%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from pandas.tseries.offsets import DateOffset

# Create a synthetic time series dataset from the property data
# Assuming the data is in a list of dictionaries format
property_data = [
    {"Price": 8025000, "Area": 1433, "Bedrooms": 3},
    {"Price": 17300000, "Area": 1408, "Bedrooms": 2},
    {"Price": 4671000, "Area": 2494, "Bedrooms": 4},
    # ... (include more data points)
]

# Convert to DataFrame
df = pd.DataFrame(property_data)

# Create a date range
date_range = pd.date_range(start='2020-01-01', periods=len(df), freq='D')

# Add the date range to the DataFrame
df['Date'] = date_range

# Set Date as index and sort
df.set_index('Date', inplace=True)
df.sort_index(inplace=True)

# Calculate daily average price
price_data = df['Price'].resample('D').mean()

# Fill NaN values with forward fill method
price_data = price_data.fillna(method='ffill')
```

```
# Fit ARIMA model
model = ARIMA(price_data, order=(1,1,1))
results = model.fit()

# Make predictions
last_date = price_data.index[-1]
future_dates = [last_date + DateOffset(days=x) for x in range(1, 31)]  # Predict next 30 days
future_datest_df = pd.DataFrame(index=future_dates, columns=['Price'])
future_df = pd.concat([price_data.to_frame(name='Price'), future_datest_df])

future_df['Forecast'] = results.predict(start=len(price_data), end=len(price_data)+29, dynamic=True)

# Visualize forecast
plt.figure(figsize=(12,6))
plt.plot(future_df.index, future_df['Price'], label='Historical')
plt.plot(future_df.index, future_df['Forecast'], label='Forecast', color='red')
plt.title('Bangalore Real Estate Price Forecast')
plt.xlabel('Date')
plt.ylabel('Average Price (INR)')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()

# Print forecast values
print(future_df['Forecast'].tail())
```

OUTPUTS AND VISUALIZATION OF THE FORCAST:




```
2020-01-29    7.240480e+06
2020-01-30    7.240480e+06
2020-01-31    7.240480e+06
2020-02-01    7.240480e+06
2020-02-02    7.240480e+06
Name: Forecast, dtype: float64
```

GitHub Link: https://github.com/FROSTYOO/3RD-PROJECT-Property_Valuation/blob/main/property_valuation.py.

STEP 5: Design Financial Equation corresponding to that Market Trend

- **Financial Equation:**

R = (P * S) - C

Where: R = Total Revenue (in Indian Rupees) P = Price per unit of service (e.g., per property valuation or per month of subscription) S = Number of sales (e.g., number of valuations performed or number of active subscriptions) C = Fixed costs (monthly operating expenses)

Let's make some assumptions based on the Bangalore real estate market and potential pricing for our service:

1. Price (P): Let's say we charge Rs. 5,000 per comprehensive property valuation report.
2. Fixed Costs (C): Assume our monthly operating expenses (including salaries, server costs, office rent, etc.) are Rs. 500,000.

Our equation becomes:

R = (5,000 * S) - 500,000

Now, let's consider different scenarios:

1. Break-even point: To find out how many sales we need to break even, we set R = 0 0 = (5,000 * S) - 500,000 500,000 = 5,000 * S S = 100 So we need to sell 100 valuation reports per month to break even.
2. Profit scenario: If we sell 200 reports in a month: R = (5,000 * 200) - 500,000 = 500,000 We would make a profit of Rs. 500,000 in this scenario.
3. Loss scenario: If we only sell 50 reports in a month: R = (5,000 * 50) - 500,000 = -250,000 We would incur a loss of Rs. 250,000 in this scenario.

This simple equation allows us to:

1. Quickly calculate expected revenue based on sales projections
2. Determine our break-even point
3. Set sales targets to achieve desired profit levels
4. Analyse the impact of pricing changes (e.g., if we change the price to Rs. 4,500 or Rs. 5,500)
5. Understand the impact of cost reductions or increases

As our business grows, we might need to adjust this model to account for:

- Variable costs that increase with each sale
- Different pricing tiers for various services
- Seasonal variations in the real estate market
- Marketing expenses that might increase to drive more sales

This financial equation provides a straightforward way to model our revenue and profitability in the Bangalore real estate market, helping us make informed decisions about pricing, sales targets, and cost management.