App Store Analysis and Recommendation System

This presentation explores the development of an intelligent recommendation system for App Store applications.

By : Rony Zareef Fayz

Agenda

Project Overview

A brief introduction to the project's goals and objectives.

App Store Data Insights

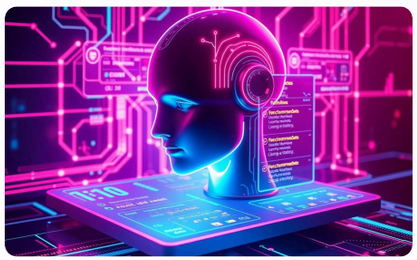
An overview of the dataset and key findings from the data analysis.

Data Preparation

The process of cleaning and preparing the data for model training.

Feature Engineering

Extracting useful information from raw data for better predictions.



Model Development

The process of developing and training a machine learning model for accurate predictions.

Recommendation System

An intelligent system that suggests relevant apps based on Content-Based Filtering.

Results and Insights

Analyzing the results of the recommendation system and deriving valuable insights.

Business Value & Future Enhancements

Exploring the business value of the recommendation system and outlining future improvements.

Project Overview

Objective

Build an intelligent recommendation system for App Store applications.

Key Focus Areas

Predict app ratings using machine learning and recommend apps based on user preferences.

Business Value

Helps users discover relevant apps faster and improves engagement.

Data Insights

Dataset	App Store data with over 2,000,000 entries.
Main Features	App details : Name, Category, Price, Size. User interaction: Rating, Rating Count, Installs . Meta data : Release Date, Minimum Android Version.
Goal	Understand patterns in ratings and suggest apps based on similar attributes.

App Store Data Distribution

Ratings

Most apps have moderate ratings (3-4 stars).

Prices

Most apps are free or have minimal price.

Installs

Visualize the distribution of app installs to understand popularity.

Data Preparation

1

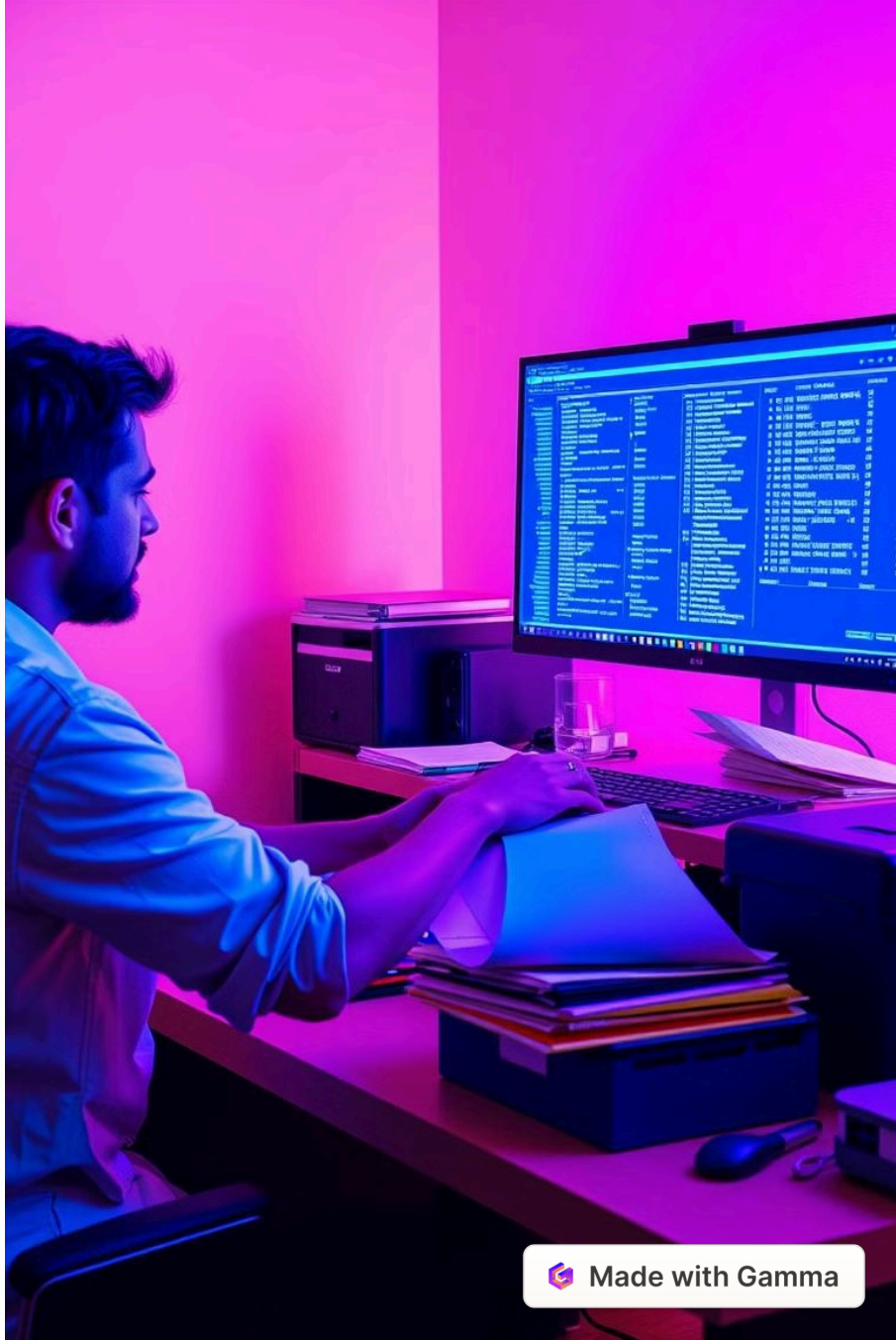
Purpose

Clean and prepare the data for model training.

2

Steps Taken

Remove missing and irrelevant data and convert features into usable formats.



Data Cleaning - Column Removal

Objective:

Enhance dataset quality by eliminating unnecessary information.

1. Removing Irrelevant Columns:

```
app.drop(['App Id', 'Developer Id', 'Developer Website',
        'Free', 'Developer Email', 'Privacy Policy',
        'Scraped Time'], axis=1, inplace=True)
```

- **Rationale:**
  - Streamlines the dataset for better analysis.
  - Reduces computational load by focusing on relevant features.

2. Benefits of Column Removal:

- Improves model training efficiency.
- Enhances interpretability of the data.

Handling Null Values

Objective:

Ensure data integrity by addressing missing values.

1. Identifying Null Values:

```
null_columns = app.isnull().sum()
```

- **Purpose:**
  - Identify columns with null values to assess data quality.

2. Removing Rows with Null Values:

```
for x in null_columns.index:
    app = app.dropna(subset=[x])
```

- **Rationale:**
  - Eliminates incomplete entries that may skew analysis results.
  - Maintains the robustness of predictive models.

Outlier Detection and Removal Using IQR Method

Objective: Identify and remove outliers from the dataset to enhance data quality and model performance.

Functionality :

1. **Count Outliers**:
   - **Calculate Quartiles**: Determines the first (Q1) and third (Q3) quartiles of the specified column.
   - **Compute IQR**: The interquartile range (IQR) is calculated as (IQR = Q3 - Q1).
   - **Identify Bounds**: Establish lower and upper bounds for outlier detection:
     - (\text{Lower Bound} = Q1 - 1.5 \times IQR)
     - (\text{Upper Bound} = Q3 + 1.5 \times IQR)
   - **Count Outliers**: Returns the number of outliers found outside these bounds.
2. **Remove Outliers**:
   - **Process Numeric Columns**: Iterates through numeric columns (excluding 'Price').
   - **Retain 'Editors Choice'**: Removes outliers while keeping entries marked as 'Editors Choice'.
   - **Output**: Returns a cleaned DataFrame devoid of outliers.

Example Usage:

```
app = remove_outliers_iqr(app)
print(app)
```

Cleaning the 'Installs' Column

Code Implementation:

```
app['Installs'] = app['Installs'].str.replace(',', '')  # Remove commas
app['Installs'] = app['Installs'].str.replace('+', '')  # Remove '+' sign
app['Installs'] = pd.to_numeric(app['Installs'], errors='coerce') # Convert to int
```

Key Steps:

- **Removing Unwanted Characters:**
  - Commas (,) and the plus sign (+) are removed to standardize values.
- **Converting to Numeric:**
  - The cleaned data is converted to a numeric format, with non-numeric entries converted to NaN (not a number).

Importance:

- Standardized values enable accurate statistical analysis and modeling, avoiding errors due to inconsistent data formats.

Cleaning the 'Size' Column

Code Implementation:

```
def convert_size(size):
    if isinstance(size, str):
        size = size.replace(',', '')  # Remove commas
        if 'M' in size:
            return float(size.replace('M', '').strip()) * 1024 * 1024  # Convert MB to bytes
        elif 'k' in size:
            return float(size.replace('k', '').strip()) * 1024  # Convert KB to bytes
        elif size == 'Varies with device':
            return None  # Handle 'Varies with device'
        else:
            return None  # Handle unexpected formats
    return None  # Handle non-string cases

app['Size'] = app['Size'].apply(convert_size)
app['Size'] = pd.to_numeric(app['Size'], errors='coerce')
```

Key Steps:

- **Defining Conversion Logic:**
  - A function is created to convert size values from strings to bytes, accounting for various formats (MB, KB, and exceptions).
- **Applying the Function:**
  - The function is applied to the Size column to standardize all size values.

Importance:

- Accurate size representation is essential for performance analysis, ensuring that app sizes are correctly quantified for insights into resource requirements.

Feature Engineering

1

2

3

Objective

Extract useful information from raw data for better predictions.

Key Features Extracted

Date-based features : Extracted year and month from release date.

Categorical encoding : Converted categories to numeric values for the model.

Business Value

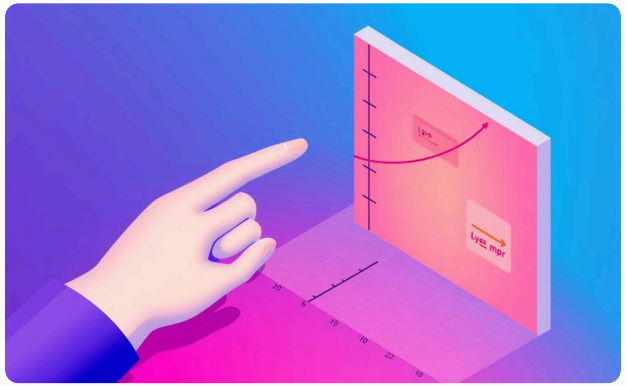
Improves the model's ability to understand patterns in the data.

Technical - Feature Engineering Code Snippet

```
# Feature extraction for date and category encoding
app['YearOfReleased'] = pd.to_datetime(app['Released']).dt.year
app['Category'], original_categories = pd.factorize(app['Category'])
```

Explanation: This code extracts the release year and encodes categories numerically.

Predictive Model (Non-technical)



Model Used: Linear Regression

Simple and easy to understand.

Why This Model?

Performs well with linear relationships.

Key Benefits

Fast training and requires less computational power.

Prediction Target: App ratings based on features such as price, category, and rating count.

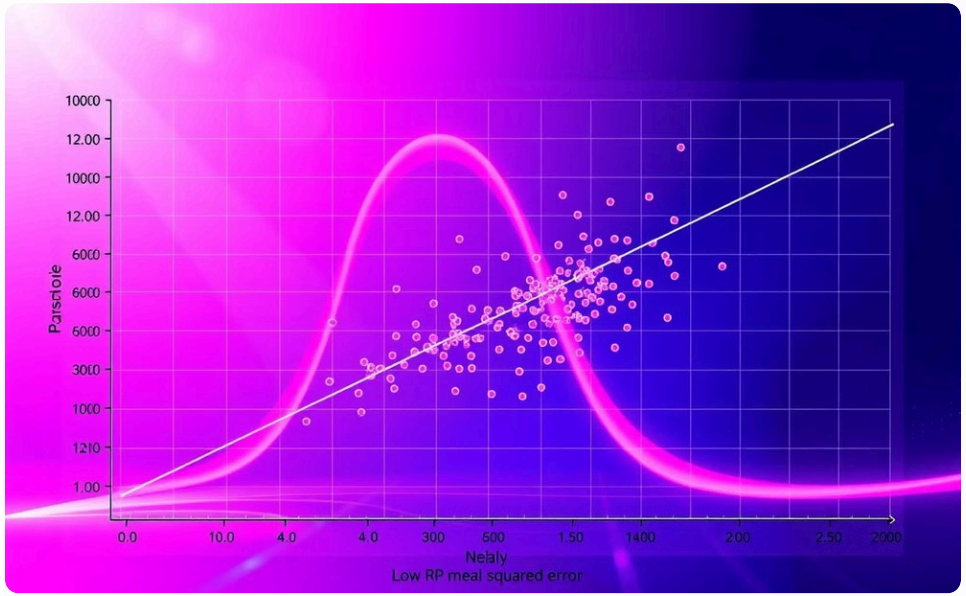
Technical - Model Training Code Snippet

```
#GridSearch dectionary
 params_lr = {
    'fit_intercept': [True, False]
}
# Perform GridSearchCV for Linear Regression
grid_search_lr = GridSearchCV(estimator=model_lr, param_grid=params_lr, cv=5, scoring='r2', verbose=1,
n_jobs=-1)
grid_search_lr.fit(x_train, y_train)
```

Explanation:

- This code trains a linear regression model on the training data to predict app ratings. The model tries to find the best linear relationship between the features and the target variable .
- Best parameters for Linear Regression: {'fit_intercept': True}.

Model Performance (Non-technical + Technical)



R-squared: Low Value

Measures how well the model predicts ratings (Score: 0.02).

Mean Squared Error (MSE): High Value

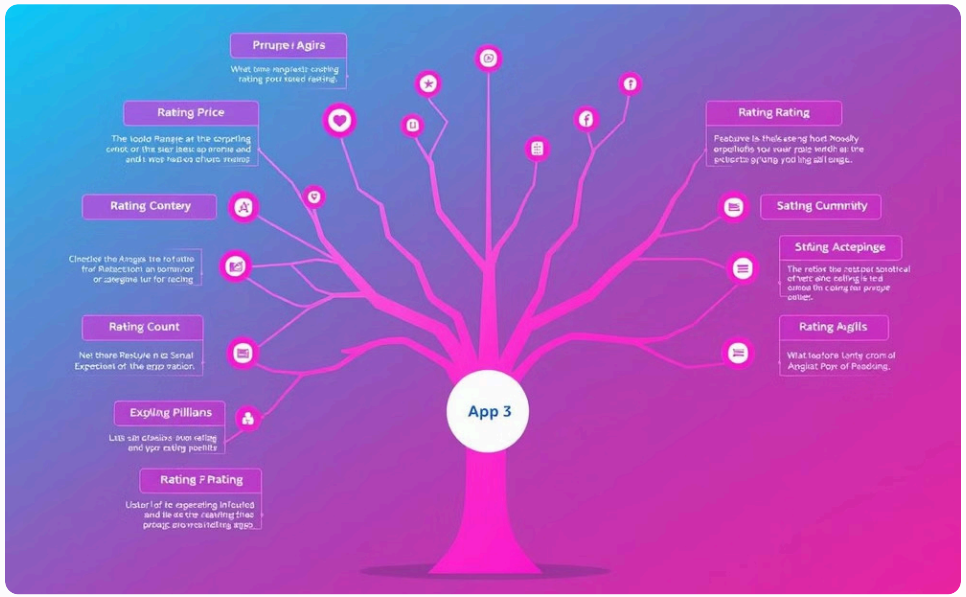
Indicates the average squared difference between predicted and actual values (Score: 4.15).

Result: The linear regression model shows limited success in predicting app ratings, with a low level of accuracy for capturing linear relationships.

Predictive Model (Non-technical)

Why Random Forest?

Handles large datasets effectively. Understands complex relationships between features.



Prediction Target

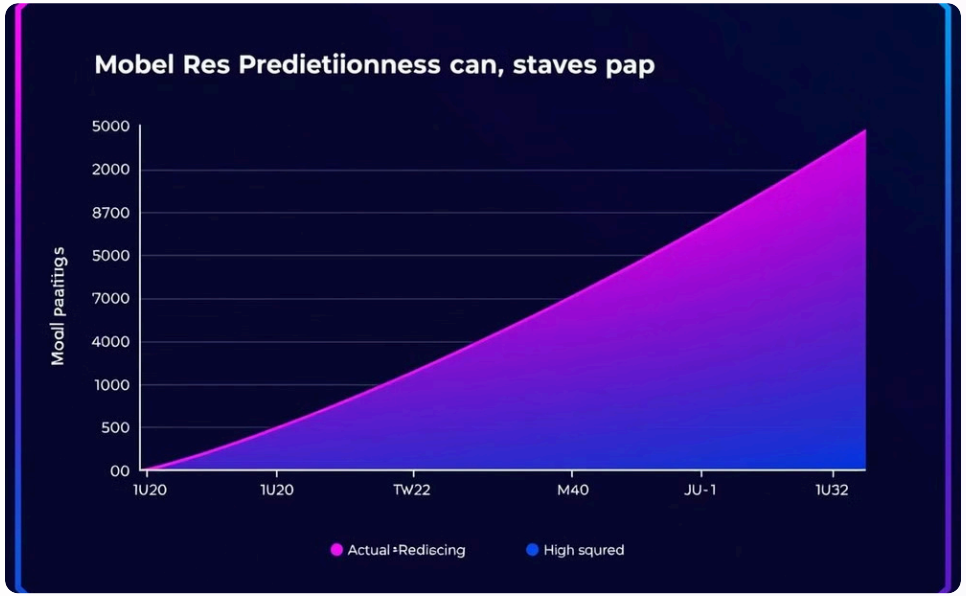
App ratings based on features like price, category, and rating count.

Technical - Model Training Code Snippet

```
#GridSearch dectionary
 params_rf = {
    'n_estimators': [50,400, 700],
    'max_depth': [None, 20, 35]
}

# Perform GridSearchCV for Random Forest
grid_search_rf = GridSearchCV(estimator=model_rf, param_grid=params_rf, cv=5, scoring='r2', verbose=1,
n_jobs=-1)
grid_search_rf.fit(x_train, y_train)
```

Explanation: Best parameters for Random Forest: {'max_depth': 20, 'n_estimators': 700} to predict app ratings.

Model Performance (Non-technical + Technical)



Evaluation Metrics

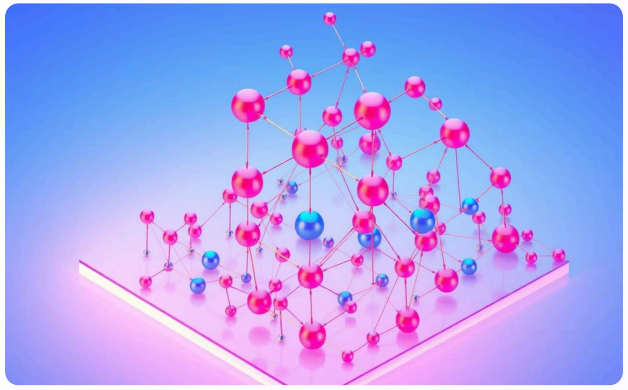
R-squared: Measures how well the model predicts ratings (Score: 0.969).

Evaluation Metrics

Mean Squared Error (MSE): Shows how close predictions are to actual values (Score: 0.05).

Result: Model successfully predicts app ratings with high accuracy.

Predictive Model (Non-technical)

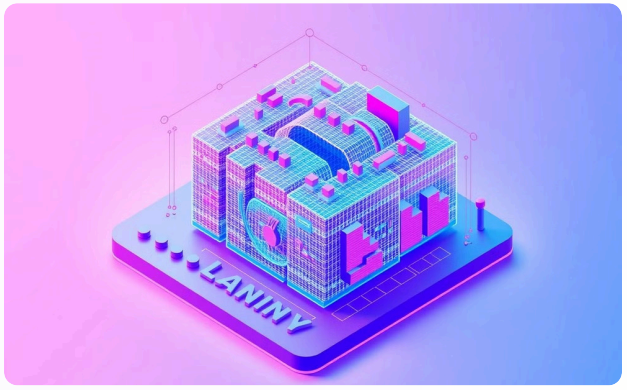


XGBoost Regressor

Combines the benefits of decision trees and boosting techniques.

Why This Model?

Highly efficient for large datasets with complex relationships.



Why This Model?

Handles missing values and outliers well.

Prediction Target: App ratings based on features such as price, category, and rating count.

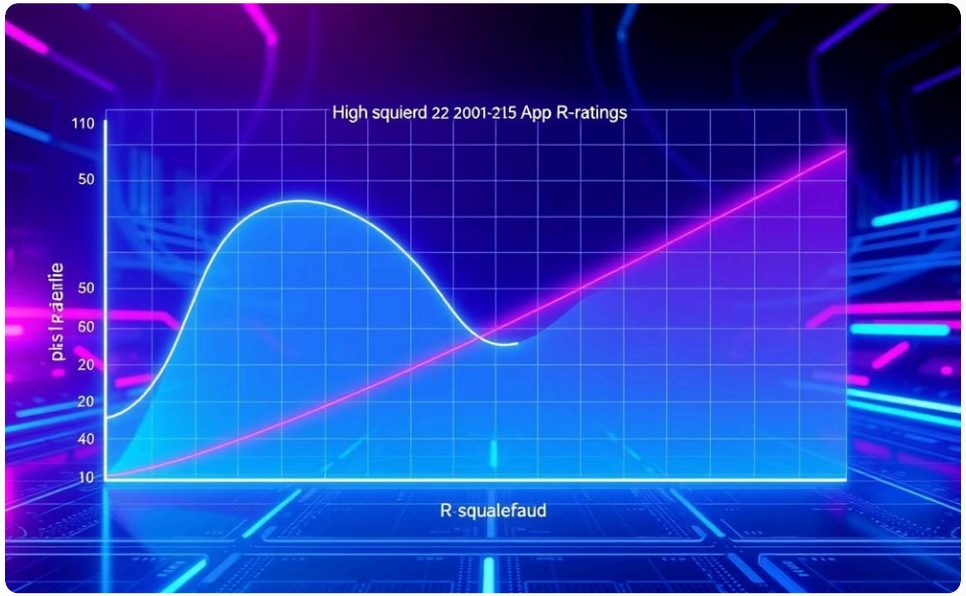
Technical - Model Training Code Snippet

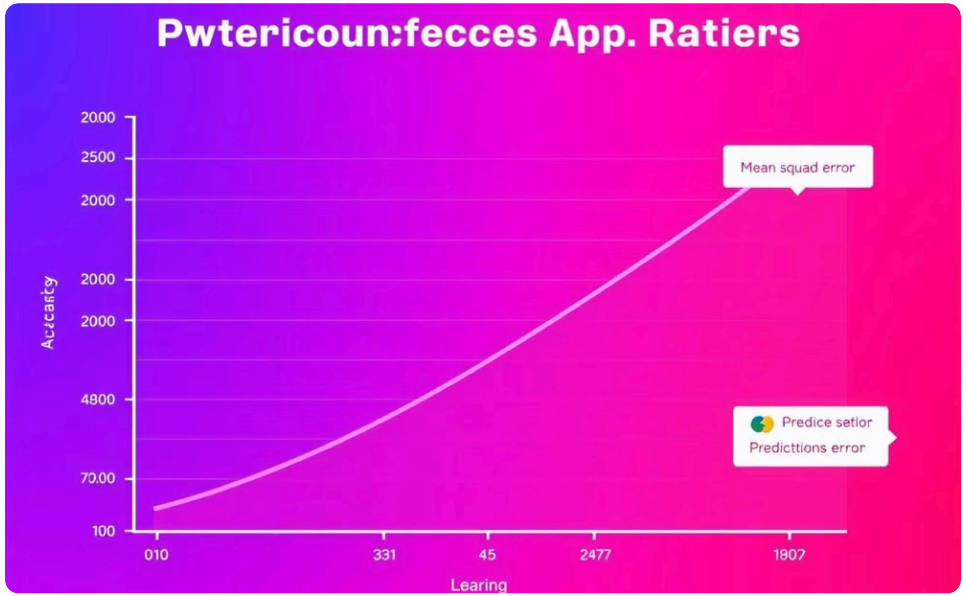
```
params_xgb = {
    'n_estimators': [50, 400, 800],
    'learning_rate': [0.01, 0.1],
    'max_depth': [10, 20, 35],
    'subsample': [0.6,1.0]
}
# Perform GridSearchCV for XGBoost
grid_search_xgb = GridSearchCV(estimator=model_xgb, param_grid=params_xgb, cv=5, scoring='r2', verbose=1, n_jobs=-1)
# Fit GridSearchCV for the model
grid_search_xgb.fit(x_train, y_train)
```

Explanation:

- This code demonstrates training an XGBoost model with Best parameters : {'learning_rate': 0.01, 'max_depth': 10, 'n_estimators': 800, 'subsample': 0.6}.
- XGBoost is known for its speed and high performance.

Model Performance (Non-technical + Technical)





R-squared

Indicates model fit and predictive power

(Score: 0.97).

Mean Squared Error (MSE)

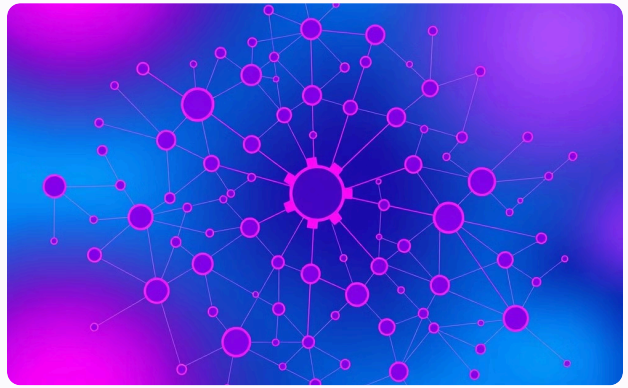
Measures prediction accuracy (lower is better), (Score: 0.12).

Result: The XGBoost model exhibits superior performance in predicting app ratings, making it a strong choice for complex data.

Predictive Model (Non-technical)

Model Used: Deep Learning (Keras)

Capable of modeling complex non-linear relationships.



Why This Model?

Flexible structure allows experimentation with layers and neurons.

Why This Model?

Effective for large and high-dimensional data.

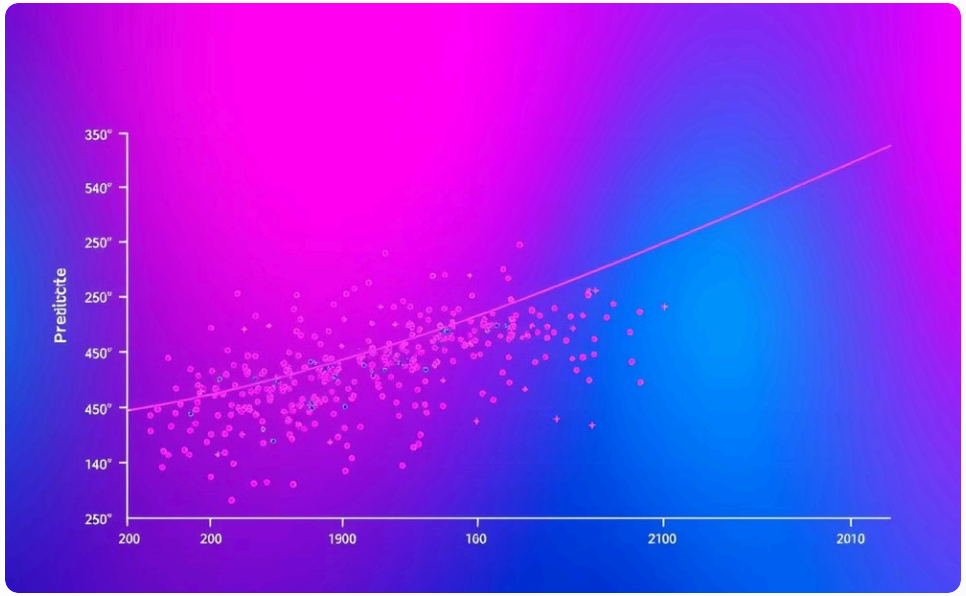
Prediction Target: App ratings based on features such as price, category, and rating count.

Technical - Model Training Code Snippet

```
from keras.models import Sequential
from keras.layers import Dense, Dropout

# Create Keras model
KerasModel = Sequential()

# Add a dense layer with ReLU activation
KerasModel.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))

# Add a Dropout layer to reduce overfitting
KerasModel.add(Dropout(0.2))

# Add output layer
KerasModel.add(Dense(1, activation='linear'))

# Compile the model
KerasModel.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_squared_error'])

# Train the model
KerasModel.fit(X_train, y_train, epochs=300, batch_size=32, validation_split=0.2)
```

Explanation: This code trains a deep learning model with a dense layer and a dropout layer to reduce overfitting. The model is compiled using the Adam optimizer and mean squared error as the loss function.

Model Performance (Non-technical + Technical)



R-squared

Reflects the model's predictive capability (Score: 0.968).

Mean Squared Error (MSE)

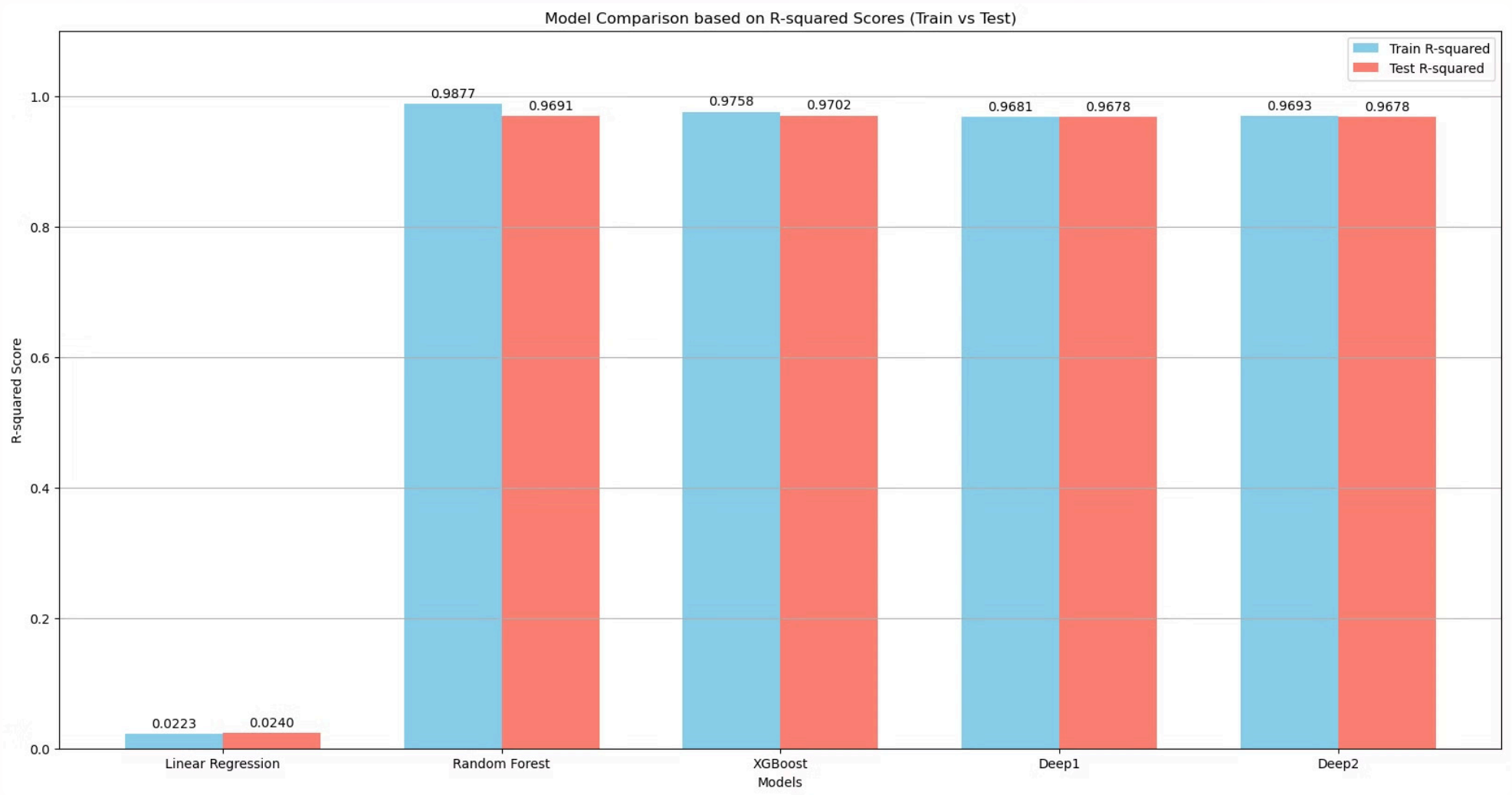
Determines the average squared difference between predicted and actual values (Score: 0.13).

Result: The deep learning model achieved high accuracy in predicting app ratings, outperforming traditional models in complex scenarios.

Model Performance Comparison

A Visual Analysis Based on R-squared Scores

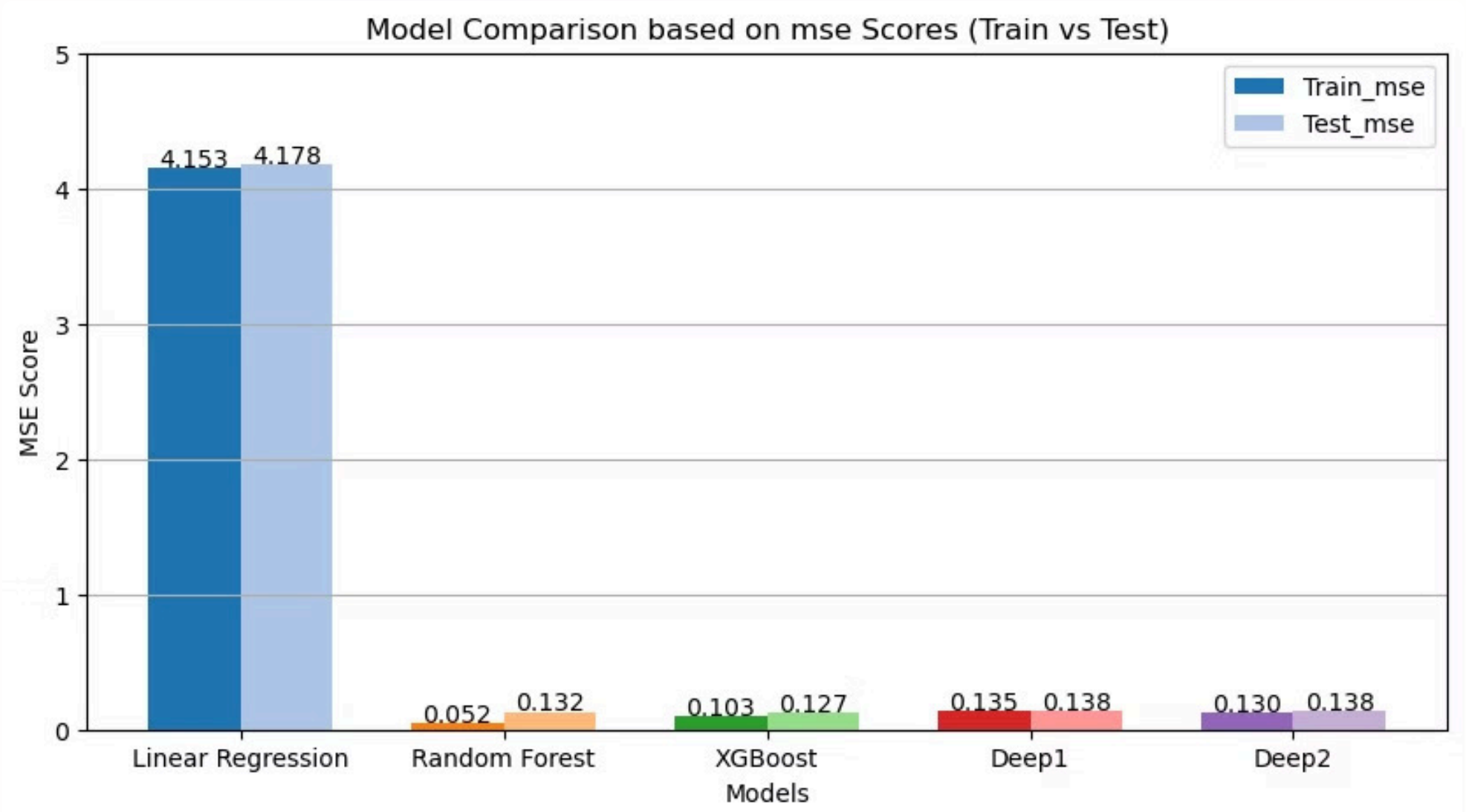
- **Image:**



- **Explanation:**
  - **Introduction:**
    - "This chart presents a comparison of various machine learning models based on their R-squared scores. R-squared is a statistical measure that indicates how well a model fits the given dataset. A higher R-squared score signifies a better fit."
  - **Key Findings:**
    - "Random Forest and Deep1 models consistently demonstrated the highest R-squared scores on both training and testing sets, suggesting they offer the best overall fit for the data."
    - "Linear Regression exhibited the lowest R-squared scores, indicating it is the least effective model for this particular dataset."
    - "A notable gap between training and testing R-squared scores for most models hints at potential overfitting. Overfitting occurs when a model becomes too specialized in the training data, compromising its ability to generalize to new, unseen data."
    - "XGBoost displayed a relatively smaller gap, suggesting it might be less prone to overfitting."

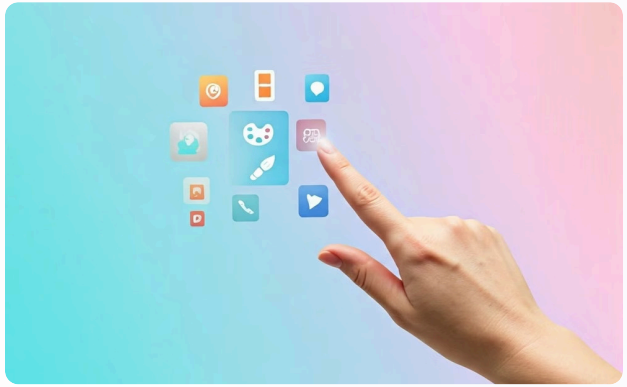
Model Performance Comparison: MSE Analysis

- **Image:**



  - **Explanation:**
    - "This chart visualizes the Mean Squared Error (MSE) of various machine learning models on both training and testing datasets. MSE is a metric that quantifies the average squared difference between the predicted and actual values."
  - **Key Findings:**
    - "Linear Regression consistently exhibits the highest MSE, indicating it's the least accurate model for this dataset."
    - "Random Forest shows a significant improvement over Linear Regression but still has room for enhancement."
    - "XGBoost demonstrates a strong performance with relatively low MSE values on both training and testing sets."
    - "Deep learning models, Deep1 and Deep2, while complex, offer comparable performance to XGBoost."
    - "A noticeable gap between training and testing MSE for most models suggests overfitting, where models become too specialized in the training data."

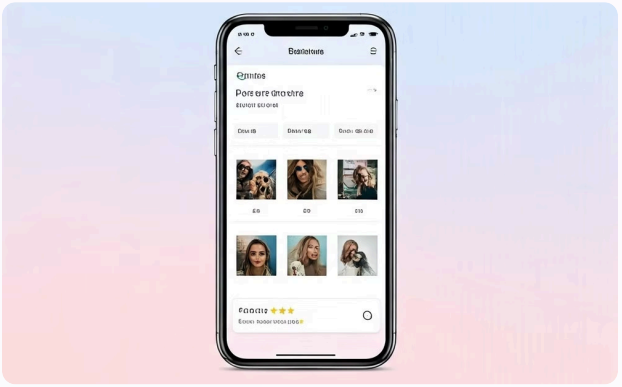
App Recommendations (Non-technical Explanation)



Approach

Suggest apps based on similarity in category, price, and

predicted ratings .



Example

If a user likes a free photo-editing app with 4+ stars, recommend similar free apps in the same category.

Business Value

Improves app discoverability and enhances user experience.

Technical - Recommendation Algorithm

```
def overall_similarity(app1, app2, category_weight=0.5):
    category_similarity = 1 if app1['Category'] == app2['Category'] else 0
    rating_similarity = 1 - abs(app1['predicted_rating'] - app2['predicted_rating']) / max(app1['predicted_rating'], app2['predicted_rating'], 1)
    price_similarity = 1 - abs(app1['Price'] - app2['Price']) / max(app1['Price'], app2['Price'], 1)
    return (category_similarity * category_weight) + (rating_similarity * 0.25) + (price_similarity * 0.25)
```

Explanation: Calculates similarity between apps based on category, predicted ratings, and price

Code: Saving Results to CSV (Technical)

```
similarities_df.to_csv('recommended_apps.csv', index=False)
```

Explanation: This code saves the top recommended apps to a CSV file.

Results

- **Example of Recommended Apps:**
  - After calculating the similarity, the system recommends the top 5 most similar apps for each of the top 30 apps.

	A	B	C	D	E
1	App	Recommended App	Category	Similarity	
2	GROW.me	shyftfly	Business	1	
3	GROW.me	Red Black Business Theme	Personalization	1	
4	GROW.me	Acadia Parish SD	Education	1	
5	GROW.me	MÃ¼ller Aktuelle Prospekt	Shopping	1	
6	GROW.me	Fraoula Mikrolimano	Food & Drink	1	
7	unlimited 4G data prank free app	Fantasy Wallpaper	Personalization	0.999592	
8	unlimited 4G data prank free app	Sunil Kumar	Personalization	0.999475	
9	unlimited 4G data prank free app	WPOR 101.9 Portland Maine Radio Free Online	Music & Audio	0.999352	
10	unlimited 4G data prank free app	Wallpapers For Robert Lewandowski Fans	Personalization	0.9992	
11	unlimited 4G data prank free app	UNLEASH 2019	Productivity	0.998998	
12	Parents	Parlamento Nacional de Timor-Leste	Events	1	
13	Parents	Easy Mixed Martial Arts Tutorial	Books & Reference	1	
14	Parents	Colour Paino Tiles	Music	1	
15	Parents	ascula	Communication	1	
16	Parents	Rice	Education	1	
17	be.MOBILISED	Space wallpapers 4K 2021	Photography	1	
18	be.MOBILISED	Premium Card |Ø§Ù„Ø¨Ø·Ø§Ù‚Ø© Ø§Ù„Ù…Ù…ÙŠØ²Ø©	Business	1	
19	be.MOBILISED	encompass KYC	Business	1	
20	be.MOBILISED	ISKCON Virtual Temples	Entertainment	1	
21	be.MOBILISED	Radio New Orleans	Music & Audio	1	
22	OTENTIK Discovery FR	ìƒëį ìŒì•…í•™ì›	Education	0.99999	
23	OTENTIK Discovery FR	Smooth Chill Radio FM UK Live Free	Music & Audio	0.99999	
24	OTENTIK Discovery FR	Drink Maker: Real Water Bootle Simulator	Casual	0.99996	
25	OTENTIK Discovery FR	Gujarat Ads	Business	0.999951	
26	OTENTIK Discovery FR	MandR	Business	0.999951	

Business Value of the Model (Non-technical)

Impact on Users

Personalized recommendations improve app discoverability.

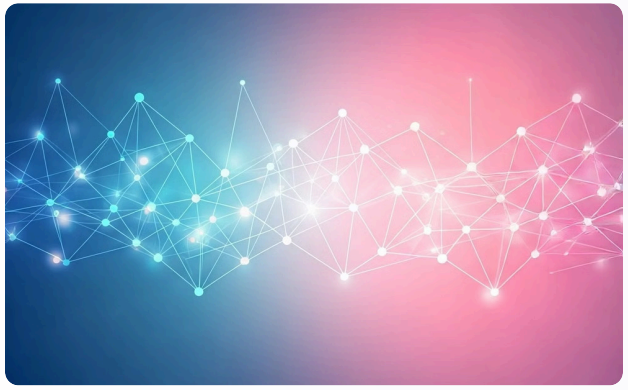
Accurate ratings help users make better decisions.

Impact on Businesses

Increased user engagement.

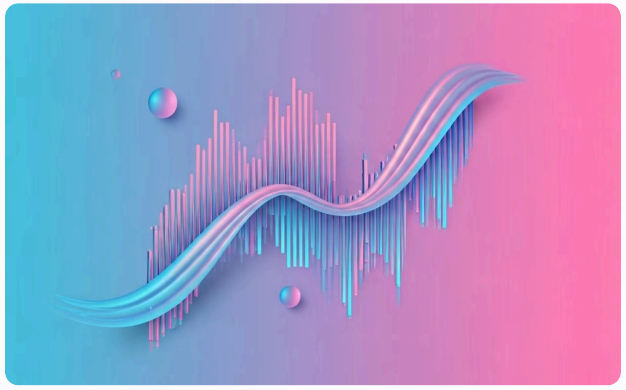
Better targeting of relevant apps for marketing efforts.

Future Enhancements (Non-technical)



Collaborative Filtering

Integrate collaborative filtering for more personalized recommendations.



Advanced Models

Apply more advanced models to improve prediction accuracy.

More Features

Expand to handle more features like user reviews and sentiment analysis.

Conclusion (Non-technical)

- **Key Takeaways:**
  - Successfully built a recommendation system using machine learning.
  - The system predicts ratings and recommends apps with high similarity.
  - Potential for further enhancements and business applications.