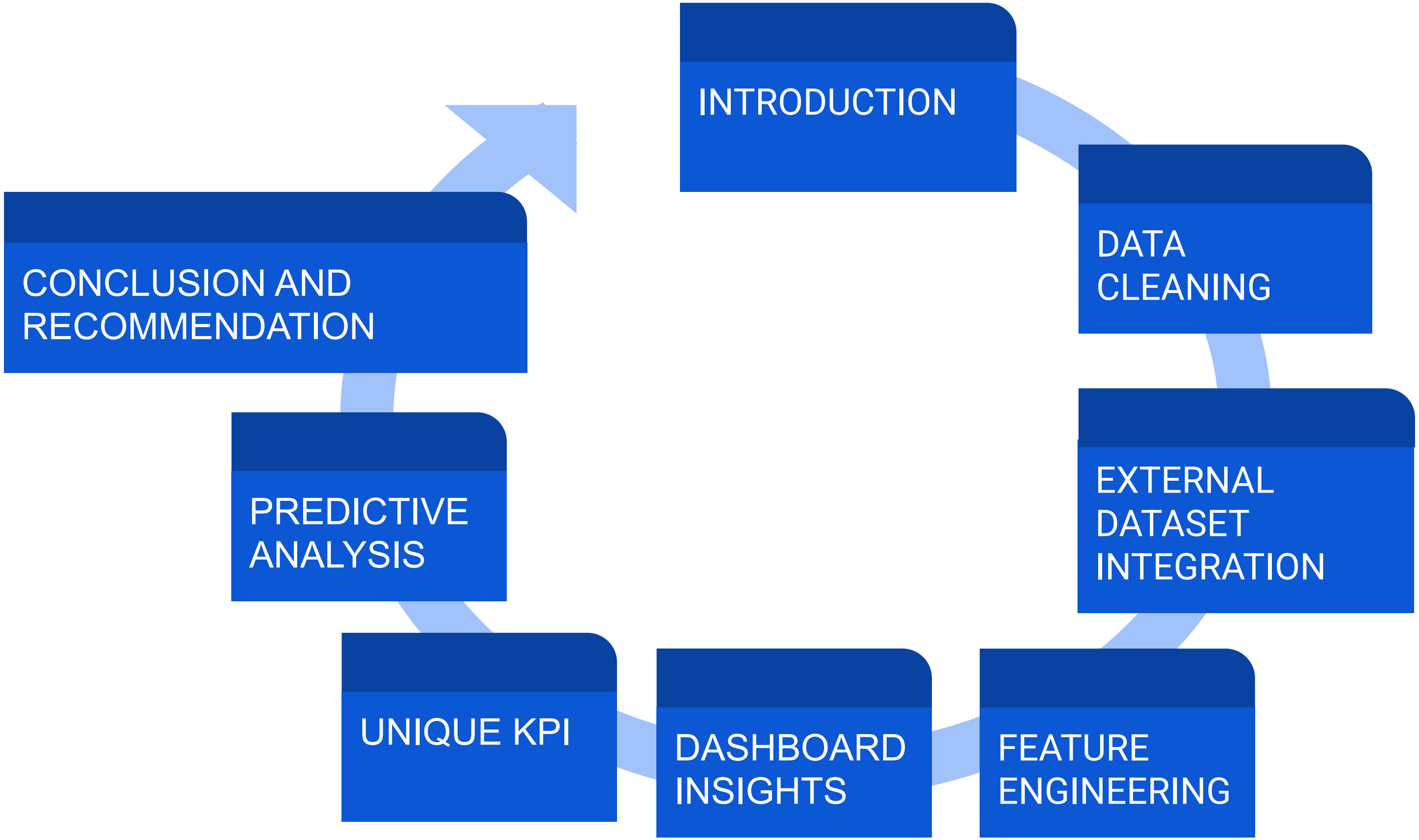
Advanced Business Insights Dashboard Challenge

C1 Senior Data Analyst Team

RELEVANT LINKS FOR THE TASK

- Google Colab Notebook
  - https://colab.research.google.com/drive/1kL8hzKPlPYH8qOwYL8pu800CueETbE0n?usp=sharing
- Microsoft power BI publish dashboard
  - https://app.fabric.microsoft.com/view?r=eyJrIjoiYTdjYTQ3MGMtNDg3Zi00MGI4LTg5YjItYzQwZTkxOTI5MzczIiwidCI6IjgwZWJlMjNkLWI0MTktNDg4Yy1iMGZjLTQ2NDhkMGI1MjQ4MSJ9
- Dashboard Documentation
  - https://drive.google.com/file/d/1VNfl6BpzuP_P8AR2-Hzrph6BYi6VAoEc/view?usp=drivesdk
- Source of External Dataset
  - https://www.kaggle.com/datasets/lava18/google-play-store-apps?resource=download

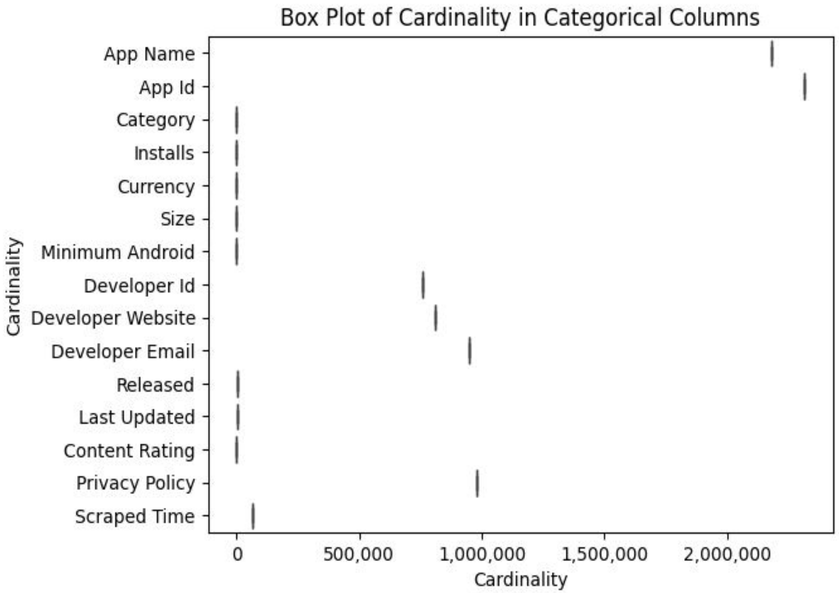
CONTENT LAYOUT



INTRODUCTION

1. Google Play Store Dataset Exploration

- ❖ The shape of the above dataset is 2,312,944 rows and 24 columns
- ❖ The datatypes comprises of 4 bool, 4 float64, 1 int64, 15 object before cleaning.
- ❖ The cardinality of the object column is been shown in the box plot below.
- ❖ The statistical view of the numerical columns are given in the table below. The count discrepancies are missing values.

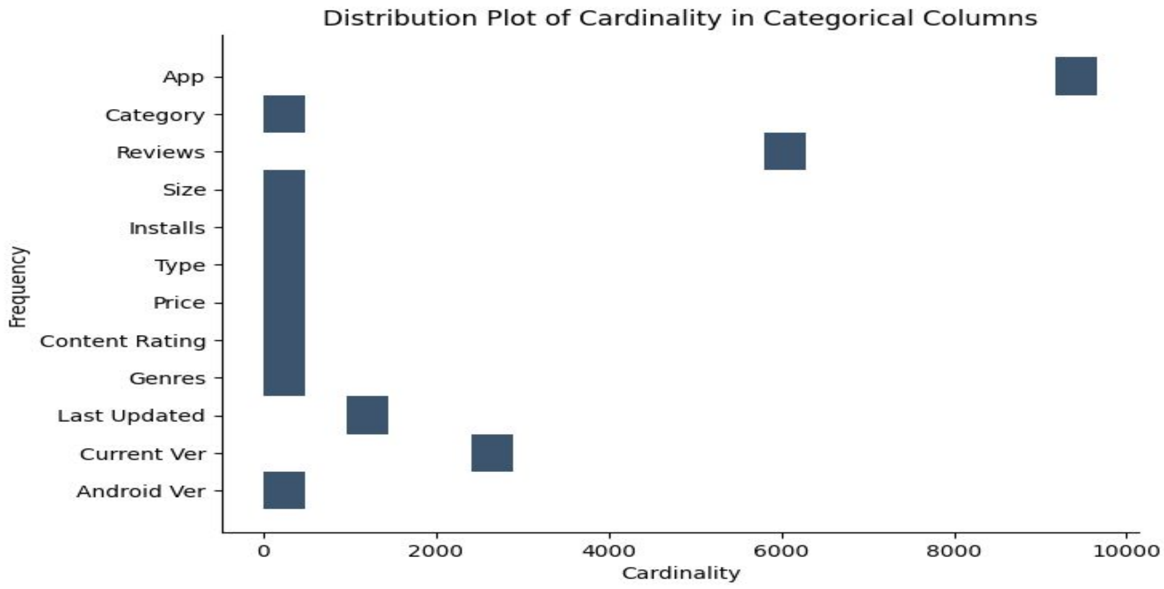


	Rating	Rating Count	Minimum Installs	Maximum Installs	Price
count	2.290061e+06	2.290061e+06	2.312837e+06	2.312944e+06	2.312944e+06
mean	2.203152e+00	2.864839e+03	1.834452e+05	3.202017e+05	1.034992e-01
std	2.106223e+00	2.121626e+05	1.513144e+07	2.355495e+07	2.633127e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00	5.000000e+01	8.400000e+01	0.000000e+00
50%	2.900000e+00	6.000000e+00	5.000000e+02	6.950000e+02	0.000000e+00
75%	4.300000e+00	4.200000e+01	5.000000e+03	7.354000e+03	0.000000e+00
max	5.000000e+00	1.385576e+08	1.000000e+10	1.205763e+10	4.000000e+02

INTRODUCTION

2. External Dataset

- ❖ As instructed, the above is an external dataset relevant to the mobile app industry
- ❖ The shape of the above dataset is 10841 rows and 13 columns
- ❖ The datatypes comprises of 1 float64 and 12 object before cleaning.
- ❖ The cardinality of the object column is been shown in the bar chart below.
- ❖ The statistical view of the float column is given in the table below.

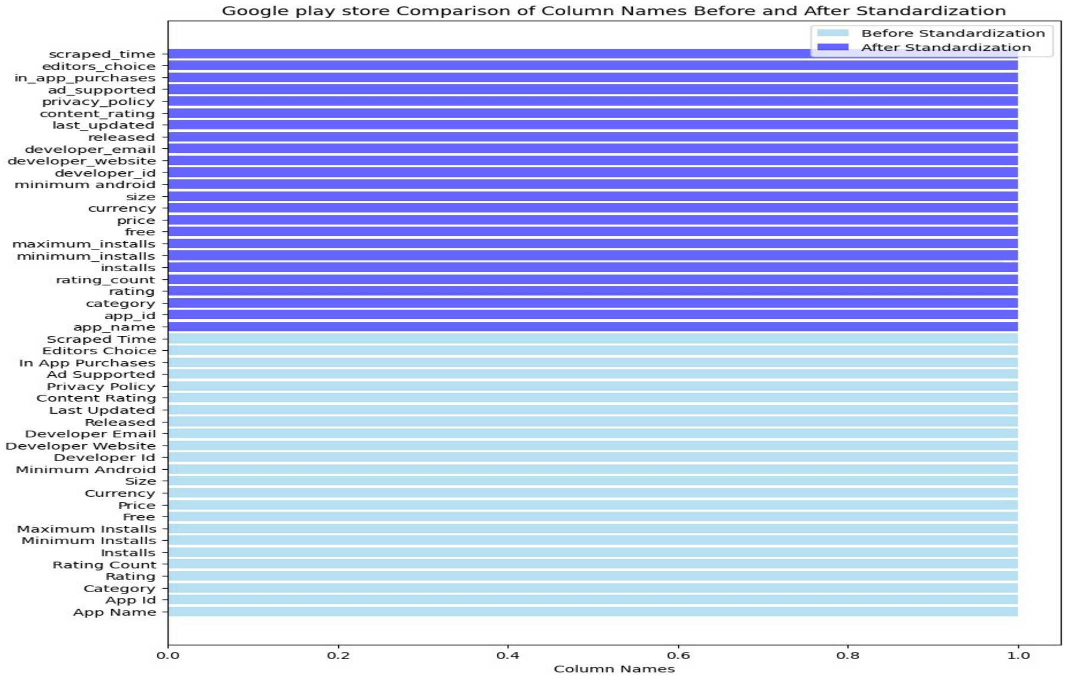


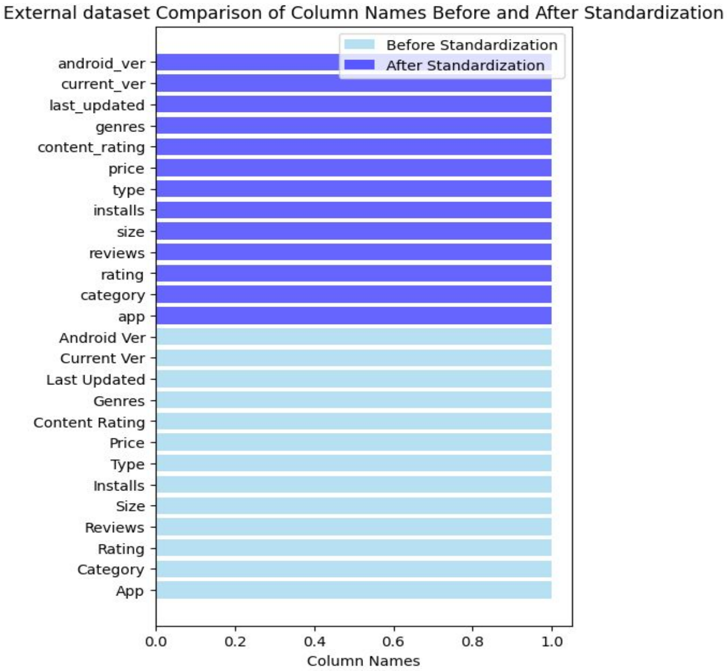
	Rating
count	9367.000000
mean	4.193338
std	0.537431
min	1.000000
25%	4.000000
50%	4.300000
75%	4.500000
max	19.000000

DATA CLEANING

1. Standardizing Data Column Names

- For the both dataset the column names follow unstandard format as seen by the sky blue bars
- The columns names was thus standardize by changing them to the lowercase using 'df.columns.str.lower() and using underscore to join words.str.replace().
- The standard column name are show with the blue bar



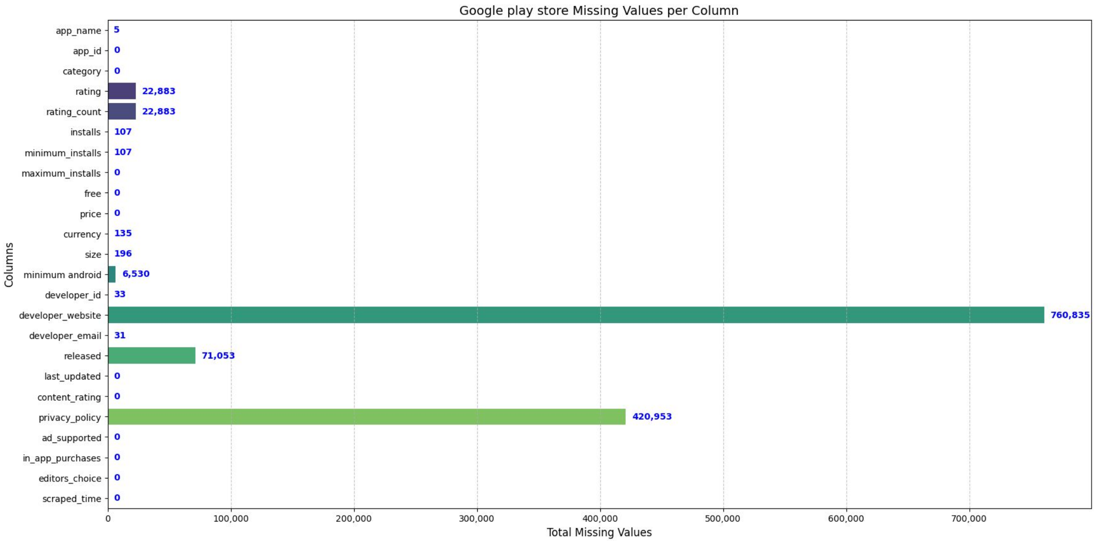


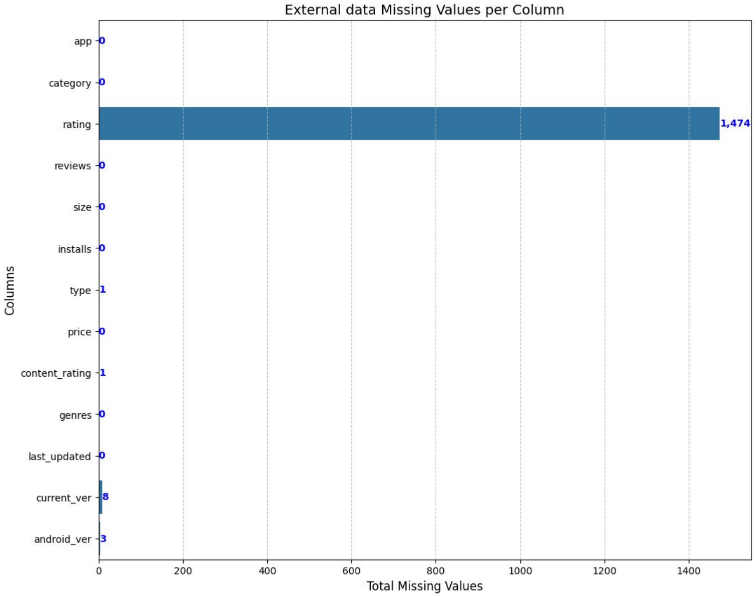
DATA CLEANING

2. Missing Values

- The total sum of missing values for the two dataset is shown in the bar plot below, with **rating and developer website** having the highest missing values in the external dataset and primary dataset.
- The approach used to clean missing values for the two dataset is dropping rows with missing values as seen in the code screenshot below

```
#Droping all the rows, app with missing values
cleaned_df = df.dropna().reset_index(drop=True)
```





DATA CLEANING

3. INVALID DATA TYPES

- 3 column for google play store dataset was changed from object to datetime as shown in the code below
- In the external dataset reviews and price was changed to numeric from object
- pd.to_numeric() for reviews and .replace().str.strip() for price.

DATA CLEANING - INVALID DATA TYPES - GOOGLE PLAY STORE

```
cleaned_df['scraped_time'] = pd.to_datetime(cleaned_df['scraped_time'])
cleaned_df['last_updated'] = pd.to_datetime(cleaned_df['last_updated'])
cleaned_df['released'] = pd.to_datetime(cleaned_df['released'])
```

10 rows of the external dataset invalid data type columns

```
cleaned_df[['reviews', 'price', 'last_updated']]
```

1 to 10 of 9360 entries Filter

index	reviews	price	last_updated
2086	38	$0.99	June 14, 2018
2087	4	$0.99	September 3, 2015
2088	38	$0.99	November 14, 2014
2159	578	$0.99	November 30, 2014
2189	178	$0.99	July 22, 2018
2212	4	$0.99	May 9, 2017
2214	66	$0.99	December 2, 2013
3262	58617	$0.99	December 7, 2016
3265	1591	$0.99	January 7, 2015
3813	12	$0.99	September 2, 2017

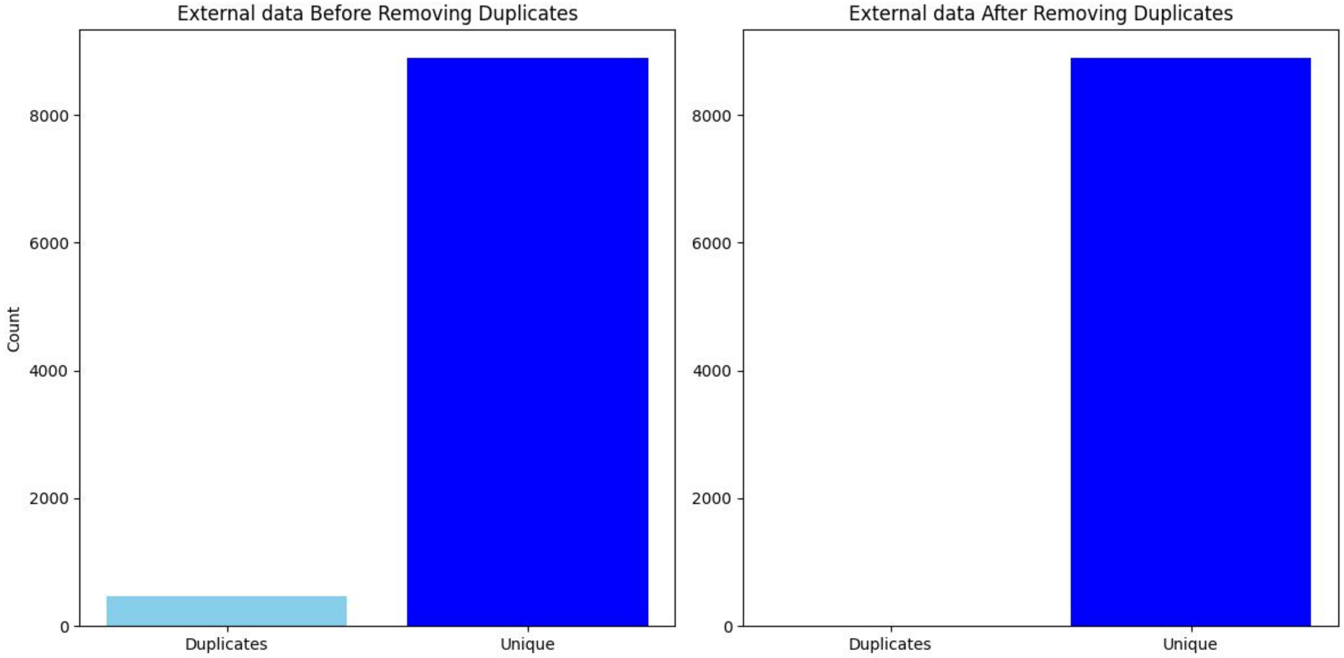
Show 10 per page

1 2 10 100 900 930

DATA CLEANING

4. Duplicate

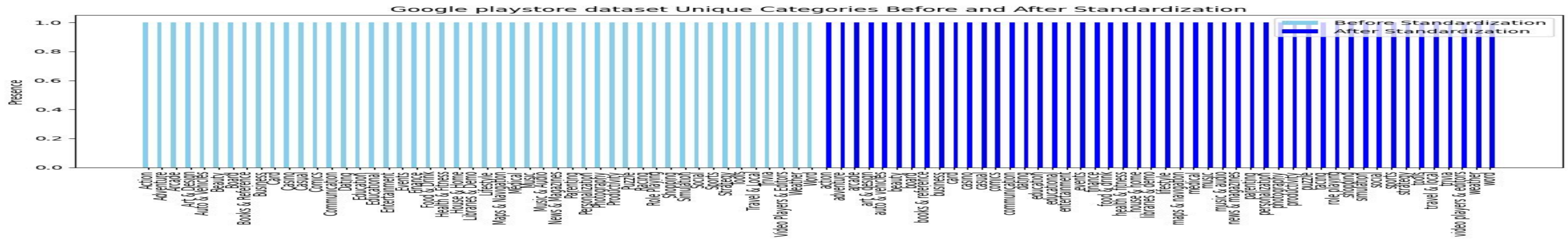
- There are 474 duplicate in the external dataset after dropping missing values in the external dataset
- Zero duplicate were found in the primary google play store dataset after dropping missing values.
- The duplication found in the external dataset as represented by the sky blue bar was droped using drop_duplicates(keep='first', inplace=True)

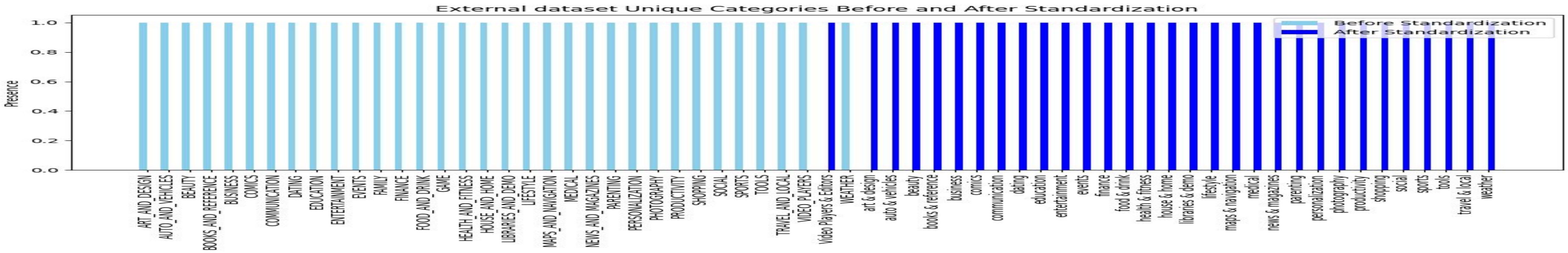


EXTERNAL DATASET INTEGRATION

1. Category Standardization Between both Dataset

- ❖ Using .str.lower() for google dataset and str.lower().str.replace('_and_', ' & ') for the external dataset. Values in the category column of both dataset were standardize.
- ❖ The light blue bar represent the row unique values before the standardization and the blue bar in the bar plot represent the row unique unique values after standardization.





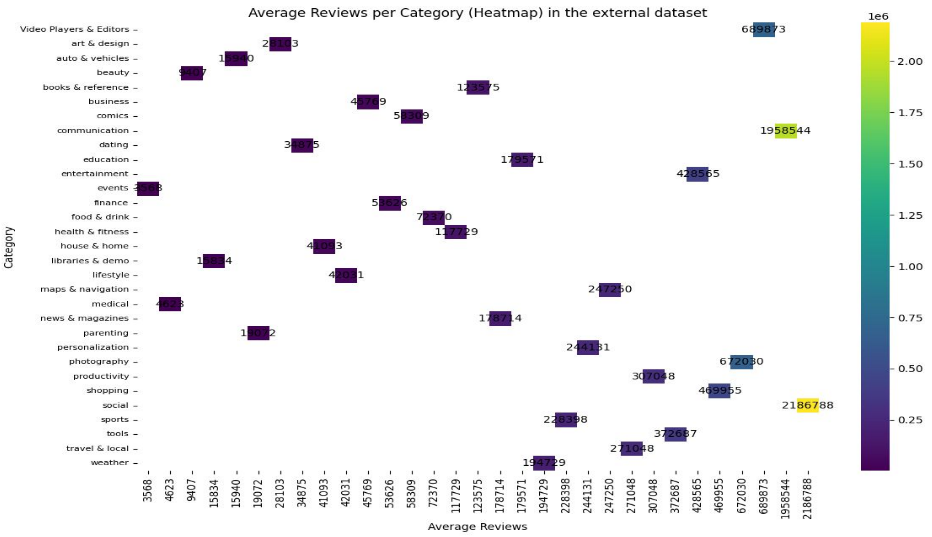
EXTERNAL DATASET INTEGRATION

2. NEW FEATURE FROM THE EXTERNAL DATASET - AVERAGE REVIEWS PER CATEGORY

```
[50] #feature engineering: average reviews per category
     cleaned_df['average_reviews_per_cat']  = cleaned_df.groupby('category')['reviews'].transform('mean').astype(int)
```

```
[51] #extract the category and average reviews, dropping duplicates to form a new dataset for the merging
     new_df = cleaned_df[['category', 'average_reviews_per_cat']].drop_duplicates()
```

- Incorporating "average reviews per category" is beneficial because user feedback and app ratings significantly influence app success in the competitive mobile app market.
- In the first code above .groupby was used to achieve this.
- The second code screenshots shows the process of dropping duplicate in category and this new feature.
- Communication and social categories have approximately 2 million average reviews per categories



EXTERNAL DATASET INTEGRATION

3. External Dataset Integration

- As shown in first code screenshot below, we merge the primary Google Play Store dataset (df2) with calculated average reviews per category from the new data frame extracted from external dataset (new_df) using a left merge on the common 'category' column.
- 250268 missing value is a result if unique categories that exist in the primary google and not the external data.
- .fillna(0) was used to fill the missing values with zero.
- 1287191 rows and 25 columns is the new shape after the integration.

```
#merging the task dataset with average reviews per category on category.
merged_df = df2.merge(new_df, on='category', how='left')
```

```
#checking for null values in the new columns -
merged_df.average_reviews_per_cat.isnull().sum()
```

```
250268
```

```
#filling those categories with 0, as we do not know their average_reviews_per_category
merged_df['average_reviews_per_cat'] = merged_df['average_reviews_per_cat'].fillna(0)
```

Merged dataset information after integration

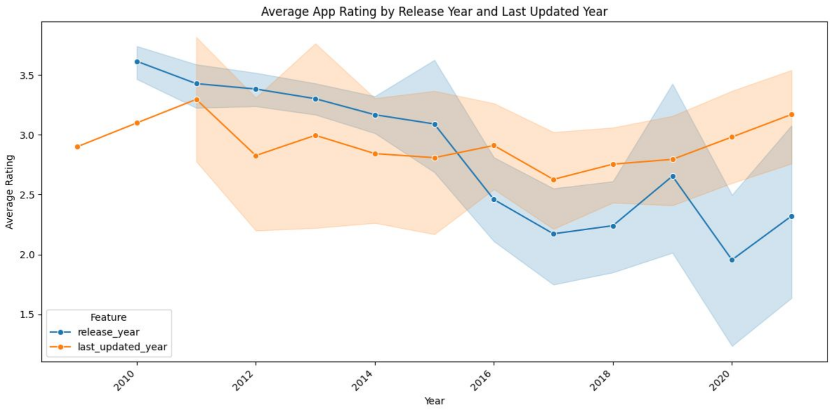
```
 #   Column                   Non-Null Count    Dtype
---  ------                   --------------    -----
 0   app_name                 1287191 non-null  object
 1   app_id                   1287191 non-null  object
 2   category                 1287191 non-null  object
 3   rating                   1287191 non-null  float64
 4   rating_count             1287191 non-null  float64
 5   installs                 1287191 non-null  object
 6   minimum_installs         1287191 non-null  float64
 7   maximum_installs         1287191 non-null  int64
 8   free                     1287191 non-null  bool
 9   price                    1287191 non-null  float64
 10  currency                 1287191 non-null  object
 11  size                     1287191 non-null  object
 12  minimum android          1287191 non-null  object
 13  developer_id             1287191 non-null  object
 14  developer_website        1287191 non-null  object
 15  developer_email          1287191 non-null  object
 16  released                 1287191 non-null  datetime64[ns]
 17  last_updated             1287191 non-null  datetime64[ns]
 18  content_rating           1287191 non-null  object
 19  privacy_policy           1287191 non-null  object
 20  ad_supported             1287191 non-null  bool
 21  in_app_purchases         1287191 non-null  bool
 22  editors_choice           1287191 non-null  bool
 23  scraped_time             1287191 non-null  datetime64[ns]
 24  average_reviews_per_cat  1287191 non-null  float64
dtypes: bool(4), datetime64[ns](3), float64(5), int64(1), object(12)
```

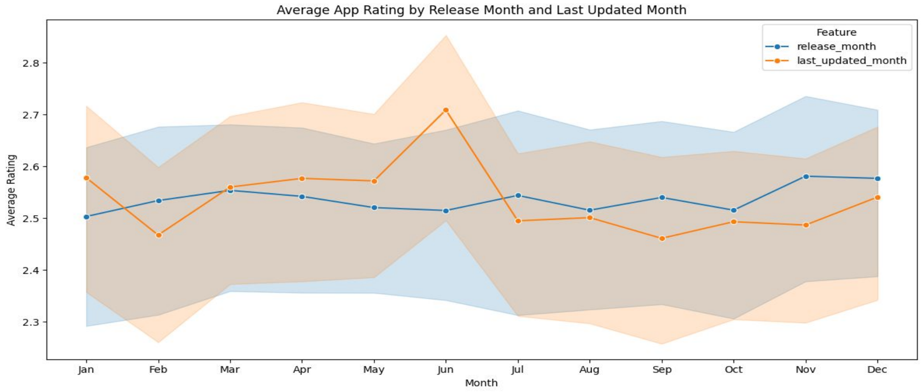
FEATURE ENGINEERING

1. Extracting Numerical Features From Datetime Columns for Predictive Purpose.

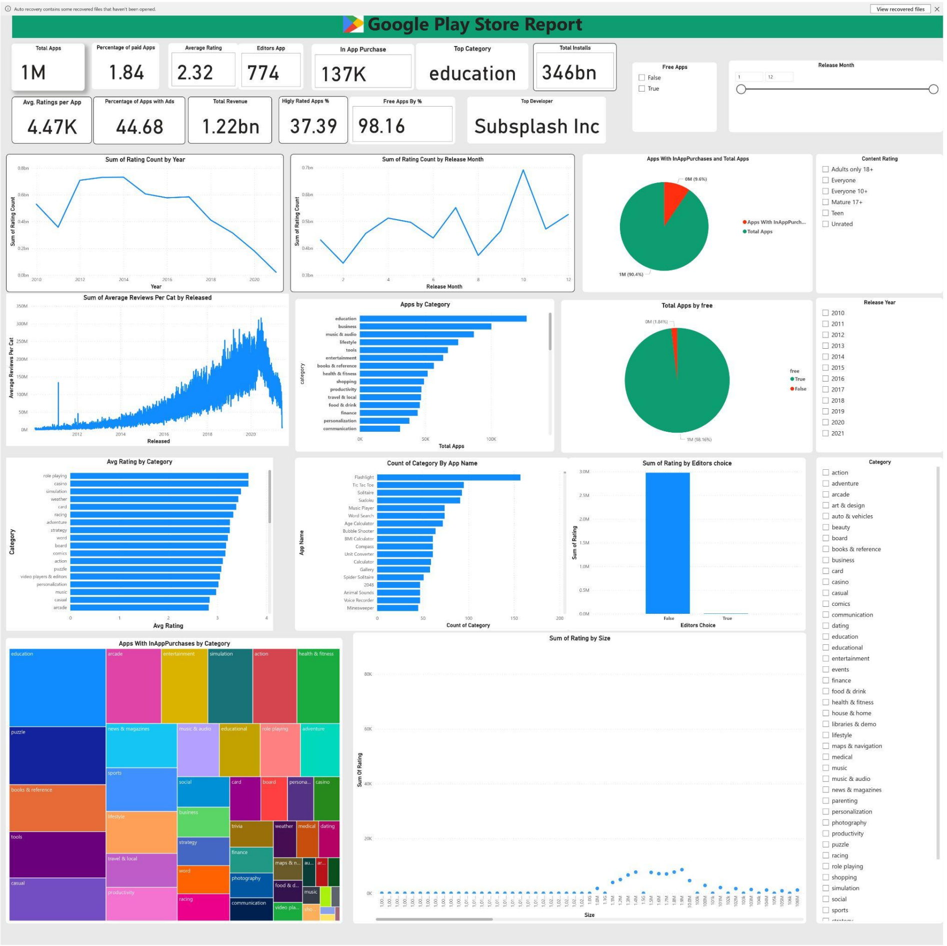
- ❖ Most machine learning algorithms based on numerical computations require input features to be in numerical format. The screenshot of the code below was used to achieve this.
- ❖ Extracting numerical features like year, month, allows machine learning models to identify temporal trends and patterns as shown in the line plots, noticing a spike in average rating on apps that are updated in july.
- ❖

```
[65] # Extract date-based numerical features
     merged_df['release_year'] = merged_df['released'].dt.year
     merged_df['release_month'] = merged_df['released'].dt.month
     merged_df['last_updated_year'] = merged_df['last_updated'].dt.year
     merged_df['last_updated_month'] = merged_df['last_updated'].dt.month
```





DASHBOARD INSIGHT



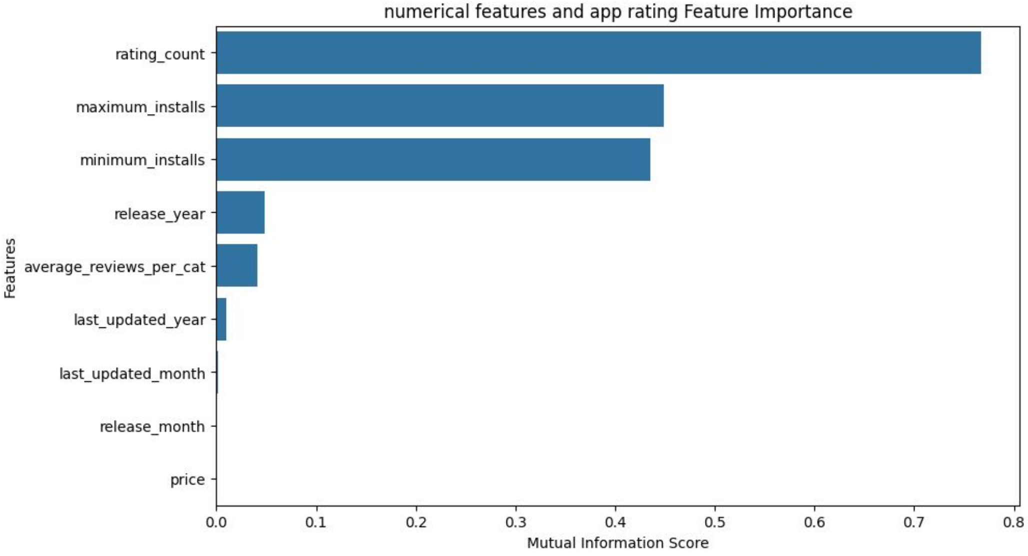
❖ INSIGHT

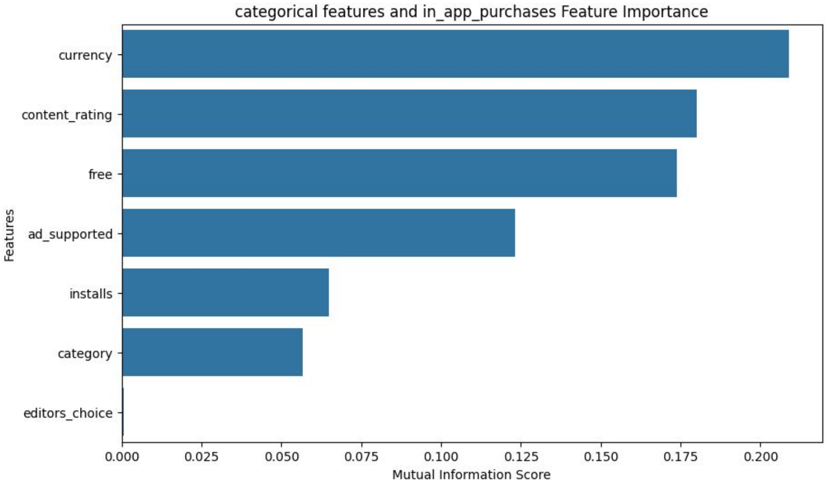
- ➢ Low Average Rating; The average rating across all apps is 2.32 indicating user dissatisfaction.
- ❖ Top category; the leading category is education.
- ❖ Total Apps; total apps hits 1 million indicating huge market with strong competition.
- ❖ Free Apps; 98.16% of free apps, monetization relies on ads or in-app purchase.
- ❖ Rating distribution over time have fluctuated significantly, with notable variance in 2011 sum of rating count was 357914073 and rose to 708231024 in 2012.

UNIQUE BUSINESS KPI

1. Features Mutual Information Score: Mutual Information measures how much knowing the value of one variable tells you about the value of the other variable, which is ideal before defining unique business KPI.

- ❖ Rating exhibits a strong relationship with user engagement metrics.
- ❖ The release year (release_year) shows a moderate association with app rating.
- ❖ For the categorical features knowing the app's currency, content rating, and whether it's free or paid provides the most information about the likelihood of it having in-app purchases.
- ❖ Apps with ads might have a tendency to also include in-app purchases. This could indicate a dual monetization strategy.





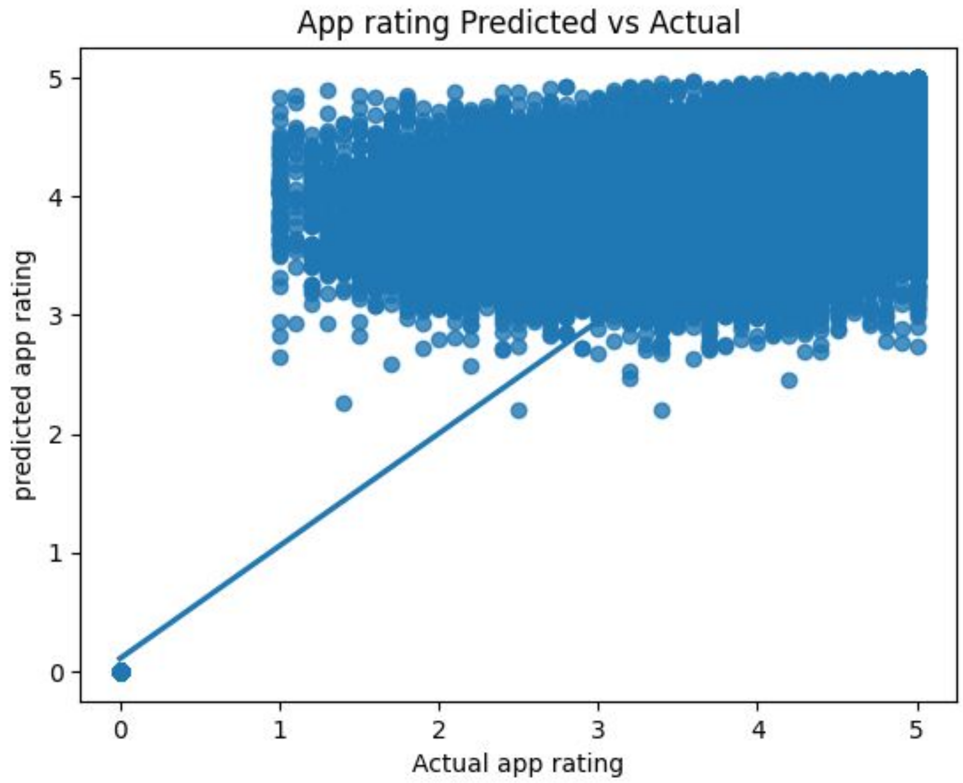
UNIQUE BUSINESS KPI

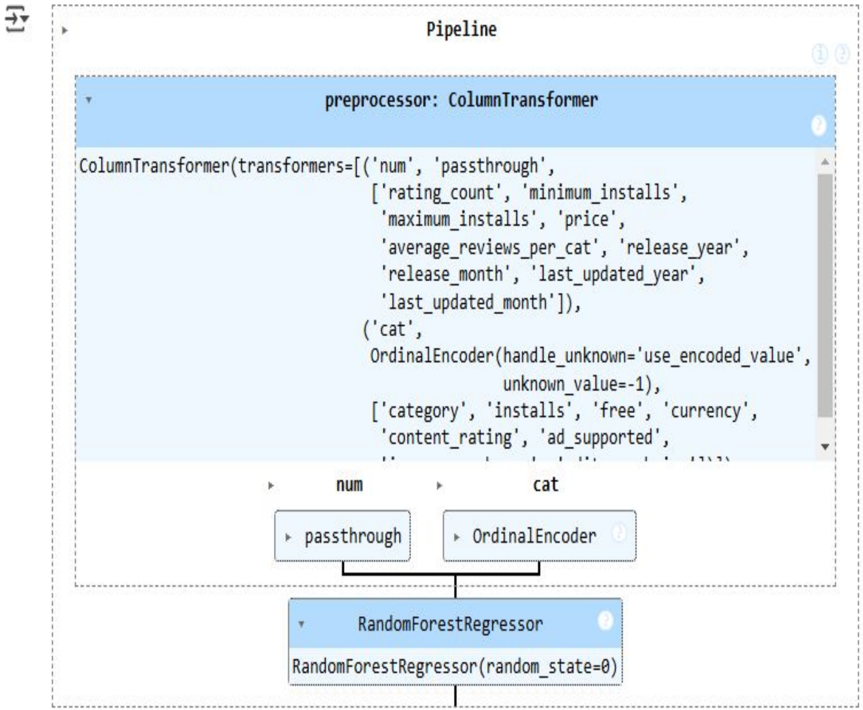
2. Unique Business KPI : Based on the features mutual information score and the Dashboard insights. The following 10 unique Business KPI has been identified;

- ❖ Percentage of Paid App
- ❖ Average rating per app
- ❖ Editors app
- ❖ In app purchase
- ❖ Top category
- ❖ Total installs
- ❖ Total Revenue : gotten from app that are not free
- ❖ Percentage of app with ads
- ❖ Free Apps By %
- ❖ Top Developer

PREDICTIVE ANALYSIS

1. Predicting Rating using Random Forest Regressor





PREDICTIVE ANALYSIS

1. Random Forest Regressor Contd

```
# Preprocessing of validation data and making predictions
prediction = my_pipeline.predict(X_valid)

# Evaluate the model
rating_score = mean_absolute_error(y_valid, prediction)
print('MAE:', rating_score)
```

```
MAE: 0.2575577772284105
```

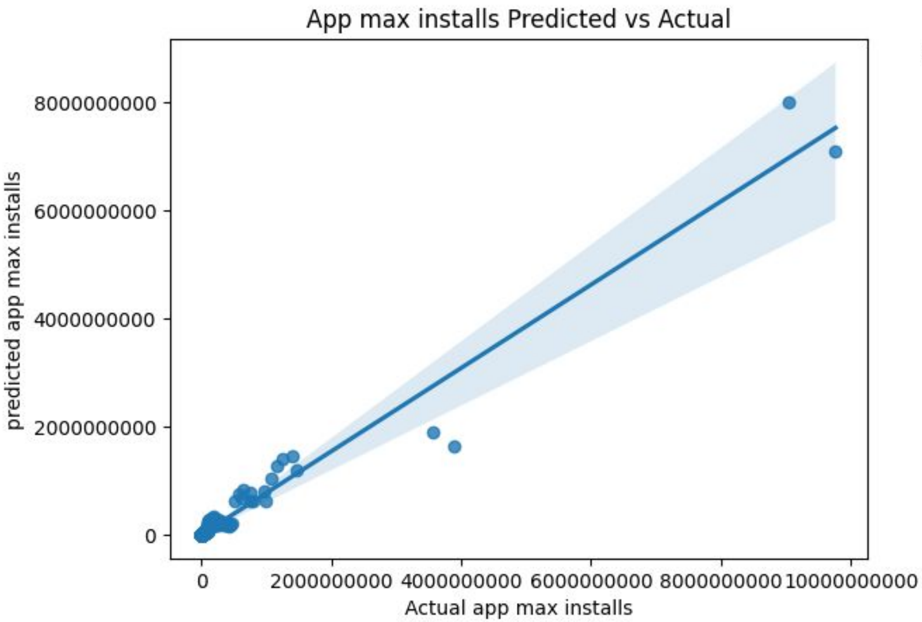
```
# Calculating R²
r2 = r2_score(y_valid, prediction)
print('R-Squared (R²):', r2)

# Calculating RMSE
rmse = np.sqrt(mean_squared_error(y_valid, prediction))
print('Root Mean Squared Error (RMSE):', rmse)
```

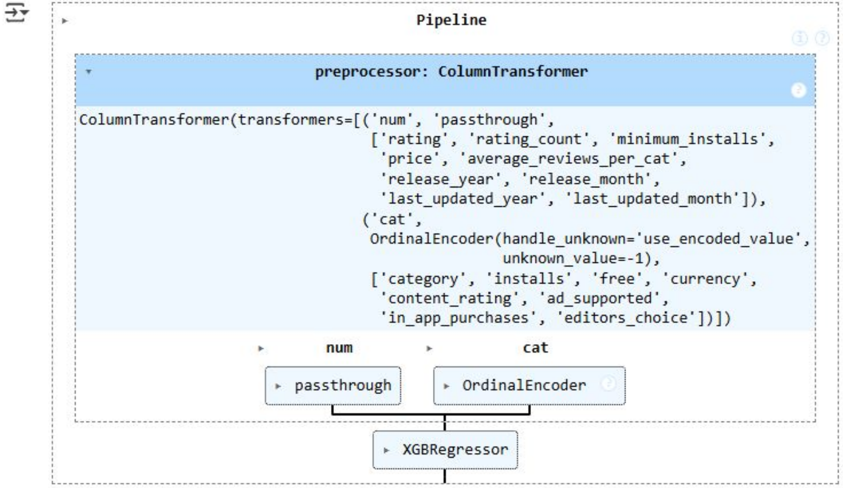
```
R-Squared (R²): 0.9509346745475418
Root Mean Squared Error (RMSE): 0.46292634811742206
```

- **Predicting App Installs**: the code Utilizes the Random forest algorithm model to predict app rating.
- **Data Preparation:** we selects relevant features (categorical with below 50 unique values and numerical , excluding datetime)
- **Pipeline for Efficiency**: it creates a streamlined workflow using a Pipeline from scikit learn, combining data preprocessing and the random forest model as shown in the screenshot above.
- **Evaluation with MAE, R^2, and RMSE** : the result of the evaluation metrics are given in the screenshot to the right
- The model explains **95%(R^2)** of the variability in app ratings suggesting a strong relationship between rating and the features.
- The model's predictions are off by an average prediction error of 0.46 rating points.
- The MAE of 0.26 smaller than the RMSE, further supporting the idea that your model is making accurate predictions
- The regplot indicates that he model has learned the features that are strongly associated with high ratings but struggle with predicting very low ratings in general.

PREDICTIVE ANALYSIS

2. Predicting maximum installs using XGBoost Regressor





PREDICTIVE ANALYSIS

1. Predicting Rating using XGBoost Regressor

```
# Train the model
my_pipeline.fit(X_train, y_train)

# Make predictions
predictions = my_pipeline.predict(X_valid)

# Evaluate the model
mae = mean_absolute_error(y_valid, predictions)
print('MAE:', mae)
```

```
MAE: 138280.96875
```

```
# Calculating R²
r2 = r2_score(y_valid, predictions)
print('R-Squared (R²):', r2)

# Calculating RMSE
rmse = np.sqrt(mean_squared_error(y_valid, predictions))
print('Root Mean Squared Error (RMSE):', rmse)
```

```
R-Squared (R²): 0.9210988283157349
Root Mean Squared Error (RMSE): 8286053.107796015
```

- **Predicting App Installs**: the code Utilizes the powerful XGBoost algorithm model to predict the maximum number of times an app will be installed.
- **Data Preparation:** Selects relevant features (categorical with below 50 unique values and numerical , excluding datetime)
- **Pipeline for Efficiency**: it creates a streamlined workflow using a Pipeline from scikit learn, combining data preprocessing and the XGBoost model as shown in the screenshot above.
- **Evaluation with MAE, R^2, and RMSE** : the result of the evaluation metrics are given in the screenshot to the right.
- On average, the model's predictions for maximum installs are off by about **138,281(MAE)** installs
- The model explains approximately **92% (R^2)** of the variance in the maximum installs.
- An RMSE of 8,286,053.11 means that, on average, the model's predictions for maximum installs are off by about 8.2 million installs
- Overall, the regplot indicates a strong positive correlation between the predicted and actual values

Conclusion

- ❖ The data cleaning and preprocessing steps significantly improved data quality, ensuring consistency in column names, handling missing values, and addressing duplicates.
- ❖ Insights from the data revealed critical trends, such as low average app ratings, dominance of the education category, and a highly competitive market with over a million apps.
- ❖ The mutual information scores highlighted key relationships between features, particularly how ratings relate to user engagement and monetization strategies.
- ❖ Despite strong predictive accuracy, challenges remain in predicting very low ratings and extreme install counts.

Recommendation

- ❖ Improving App Ratings: Developers should focus on enhancing user experience, addressing negative feedback, and optimizing app performance to improve average ratings.
- ❖ Strategic Monetization: Given the high percentage of free apps, businesses should explore in-app purchases and ads as revenue models while balancing user satisfaction.
- ❖ Targeted Category Growth: Since the education category is leading, app developers can leverage this trend by creating high-quality educational apps.
- ❖ Enhancing Predictive Models: Further fine-tuning and incorporating additional features may help improve model accuracy, particularly in predicting low ratings and extreme install numbers.