# Data Analytics and Visualization on Fiverr Dataset

## Data Cleaning and Exploratory Data Analysis (EDA) on Fiverr Freelance Dataset

**Manahil Ahmad**

# Table of Content

# 1. Introduction

This report presents a comprehensive data cleaning and exploratory data analysis (EDA) on the Fiverr Freelance Dataset.
The dataset was analyzed using Python libraries such as pandas, numpy, matplotlib, scipy, and pathlib to clean raw data, handle missing values, remove outliers, and visualize key insights related to Fiverr gig prices, reviews, and category trends.
The purpose of this analysis was to identify:

- Which Fiverr categories have higher average prices.
- The correlation between gig prices, ratings, and review counts.
- The most profitable and in-demand skill categories.

# 2. Dataset Description and Justification

**Dataset Name:** fiverr.csv
**Source:** Collected from Fiverr category listings.
**Records:** 6,183 rows
**Attributes:** 7 columns

| Column | Description |
|---|---|
| Category | Main Fiverr service category (e.g., Programming & Tech, Design, Writing). |
| Subcat | Subcategory of the service. |
| name | Title/description of the gig. |
| stars | Text-formatted rating with review count (e.g., 5.0(1k+)). |
| price | Starting price of the gig (e.g., Starting at €10). |
| Category-href, Subcat-href | URLs for category pages. |

**Justification:**
The Fiverr dataset offers valuable insights into the freelance marketplace. It was selected for analysis due to its clear structure, real-world application, and inclusion of both numeric and categorical attributes.

# 3. Data Pre-processing

### Step 1: Import Libraries and Load Dataset

**Code:**

```
import numpy as np
import pandas as pd
file_path = "/content/drive/MyDrive/fiverr.csv"
df=pd.read_csv(file_path)
print("Dataset Loaded Successfully!")
```

**Observation:**
Dataset loaded successfully with **6,183 rows and 7 columns**.

## Step 2: Initial Data Exploration

**Code:**

```
df.info()
print("Shape:", df.shape)
print("\nColumns:", df.columns.tolist())
df.head()
df.describe()
df.isnull().sum()
```

**Purpose:** To understand dataset structure, data types, and missing values.
**Observation:**

- Some missing values in stars.
- All columns were string type (object).

## Step 3: Clean Column Names

**Code:**

```
#Remove leading/trailing spaces from string cells
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].astype(str).str.strip()
```

**Purpose:**
Standardize column names and remove extra spaces.
**Result:**
All column names formatted consistently (e.g., Category-href → category_href).

**Code:**

```
# Clean 'stars' column (extract rating + review count)
df['stars'] = df['stars'].fillna('').astype(str).str.strip()
df['star_rating'] = df['stars'].str.extract(r'(\d+(?:\.\d+)?)').astype(float) df['review_raw']
= df['stars'].str.extract(r'\(([^)]+)\)')[0]
```

**Purpose:**
Extract numeric ratings and review counts.

**Observation:**
New columns created:

- star_rating → numeric gig rating
- review_count → number of reviews (converted from text such as “1k+” to 1000).

## Step 4: Remove Duplicates

**Code:**

```
duplicates = df.duplicated().sum()
```

```
print(f"Duplicate rows found: {duplicates}")
df = df.drop_duplicates()
print("Duplicates removed.")
```

**Observation:**
Duplicate rows found and removed — ensuring each Fiverr gig is unique.

## Step 5: Handle Missing Values

**Code:**

```
print("\nMissing values before cleaning:\n", df.isnull().sum())
threshold = 0.35
df = df.loc[:, df.isnull().mean() < threshold]
for col in df.select_dtypes(include='object').columns:
    df[col].fillna(df[col].mode()[0], inplace=True)
for col in df.select_dtypes(include=np.number).columns:
    df[col].fillna(df[col].median(), inplace=True)
print("\nMissing values after cleaning:\n", df.isnull().sum())
```

**Observation:**

- Columns with more than 35% missing data were dropped.
- Remaining missing values were filled (categorical with mode, numeric with median).

```
Missing values before cleaning:
 category          0
category-href     0
subcat            0
subcat-href       0
name              0
stars             0
price             0
star_rating     493
review_raw      493
review_count    493
dtype: int64

Missing values after cleaning:
 category        0
category-href   0
subcat          0
subcat-href     0
name            0
stars           0
price           0
star_rating     0
review_raw      0
review_count    0
dtype: int64
```

## Step 6: Clean and Convert Price Column

**Code:**

```
def extract_price(x):
```

```
    if pd.isna(x):
        return np.nan
    x = str(x)
    # Remove currency symbols and letters
    x = re.sub(r'[^0-9\.\-\s]', '', x)
    # Handle ranges like "10-20" or "10 20"
    if '-' in x:
        parts = [float(p) for p in x.split('-') if p.strip()]
        return np.mean(parts) if parts else np.nan
    elif ' ' in x:
        parts = [float(p) for p in x.split(' ') if p.strip()]
        return np.mean(parts) if parts else np.nan
    try:
        return float(x)
    except:
        return np.nan
df['price_num'] = df['price'].apply(extract_price)
df = df[df['price_num'].between(5, 5000)]
print("\nPrice column cleaned and converted to numeric.")
df[['price', 'price_num']].head()
```

**Observation:**

- Extracted numeric price values.
- Removed outliers (prices <5€ or >5000€).

## Step 7: Convert Stars and Reviews to Numeric

**Code:**

```
for col in ['stars', 'reviews', 'ratings']:
    if col in df.columns:
        # Extract first numeric value (integer or decimal) using regex
        df[col] = pd.to_numeric(
            df[col].astype(str).str.extract(r'(\d+\.?\d*)')[0],
            errors='coerce'
        )
df.head()
```

**Purpose:**
Convert text ratings/reviews into numeric values for correlation analysis.

## Step 8: Outlier Detection and Treatment

**Code:**

```
numeric_cols = [price_col, reviews_col]  # you can also add stars_col if numeric
for col in numeric_cols:
    if col in df.columns:
        # Remove NaNs
        df = df[df[col].notna()]
        # Calculate IQR
```

```
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        # Define bounds
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        # Count outliers
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
        print(f"    {len(outliers)} outliers detected in '{col}'")
        # Remove them
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
print("Outliers removed. Proceeding to correlation analysis.")
print(f"Remaining rows: {len(df)}")
```

**Observation:**
Outliers in price and review_count were detected and removed using the **IQR method** to improve data consistency.

### Step 9: Inspect and Save Cleaned Dataset

**Code:**

```
print("Final Shape:", df.shape)
print("Data Types:\n", df.dtypes)
print("\nSample Data:\n", df.head())
cleaned_path = "/content/drive/MyDrive/fiverr_cleaned.csv"
df.to_csv(cleaned_path, index=False)
print(f"Cleaned dataset saved to: {cleaned_path}")
```

**Observation:**
Cleaned dataset successfully saved for visualization and further analysis.

### Step 10: Save Cleaned Dataset

Succesfully saved the cleaned dataset.

## 4. Exploratory Data Analysis (EDA)

### Step 11: Identify Key Columns for Analysis

```
def find_col(df, names):
    for n in names:
        for c in df.columns:
            if n.lower() in c.lower():
                return c
    return None

price_col = find_col(df, ["price_num", "price"])
cat_col = find_col(df, ["category", "subcat"])
stars_col = find_col(df, ["stars", "rating"])
```

```
reviews_col = find_col(df, ["reviews", "review_count"])

print("Using columns → Price:", price_col, "| Category:", cat_col)
```

**Observation:**
Dynamic column detection ensures analysis runs correctly regardless of naming differences.

## Step 12: Average Price by Category

**Code:**

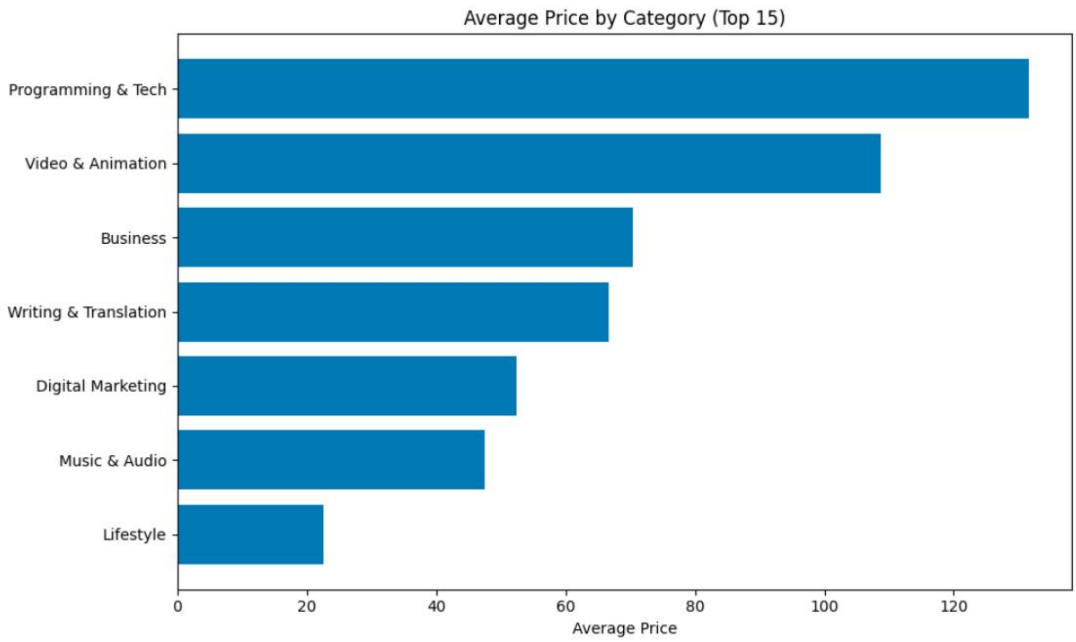
```
avg_price = df.groupby(cat_col)[price_col].agg(['count', 'mean',
'median']).reset_index()
avg_price = avg_price.sort_values('mean', ascending=False)
print("\nTop Categories by Average Price:\n")
print(avg_price.head(10))
```

**Observation:**
Identified **top Fiverr categories** by average gig price — technical categories lead, followed by business and design.

## Step 13: Bar Chart – Top 15 Categories

**Code:**

```
top15 = avg_price.head(15).sort_values('mean')
plt.figure(figsize=(10,6))
plt.barh(top15[cat_col], top15['mean'])
plt.xlabel("Average Price")
plt.title("Average Price by Category (Top 15)")
plt.tight_layout()
plt.show()
```



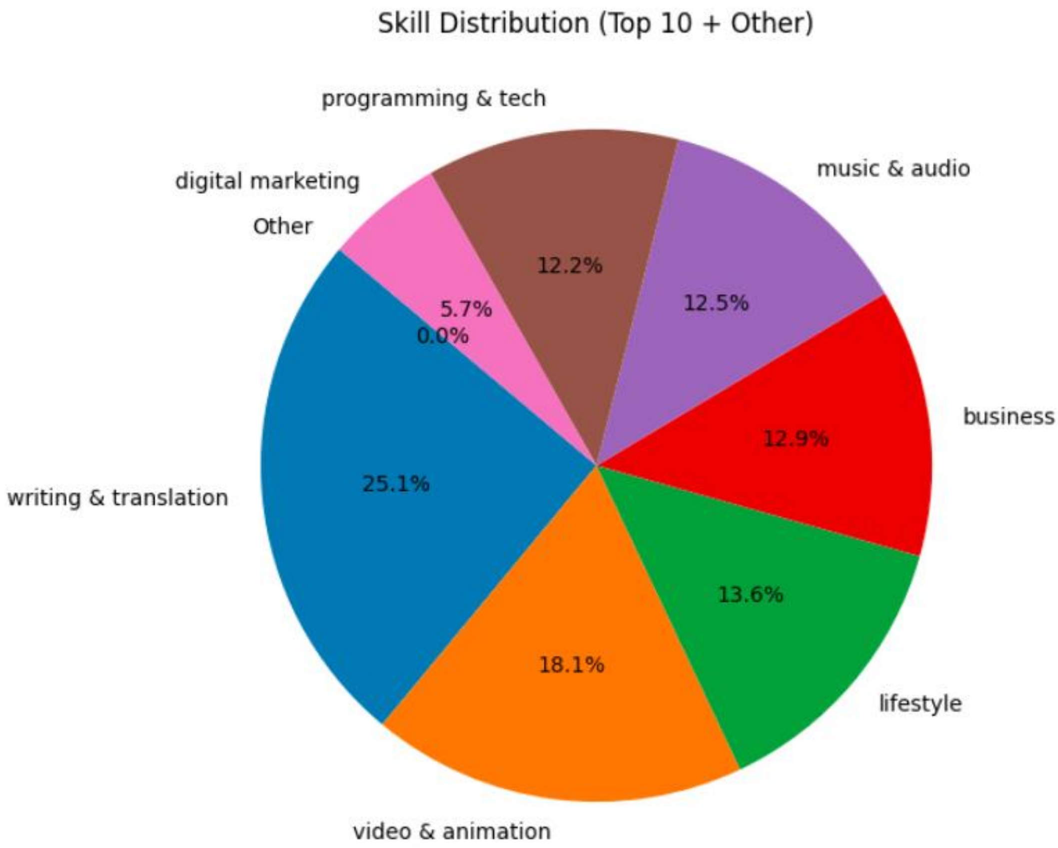

**Result:**
Visualized the top 15 categories with highest average gig prices.

## Step 14: Skill Distribution (Pie Chart)

**Code:**

```
top10 = skill_freq.head(10)
others = skill_freq['count'].sum() - top10['count'].sum()

plt.figure(figsize=(7,7))
plt.pie(
   top10['count'].tolist() + [others],
   labels=top10['skill'].tolist() + ['Other'],
   autopct='%1.1f%%',
   startangle=140
)
plt.title("Skill Distribution (Top 10 + Other)")
plt.show(
```




)

**Observation:**
Pie chart shows top 10 most frequent skills/categories on Fiverr.

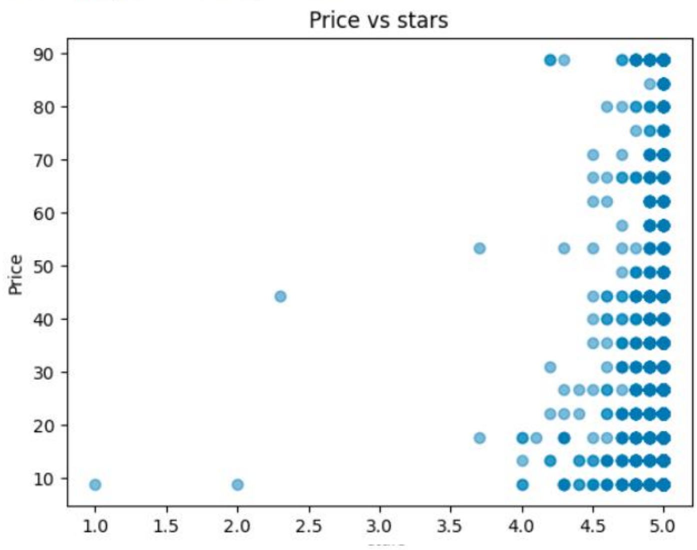
## Step 15: Correlation – Price vs Stars/Review

**Code:**

```
df[price_col] = pd.to_numeric(df[price_col], errors='coerce')
if stars_col:
```

```
    df[stars_col] = pd.to_numeric(df[stars_col], errors='coerce')
    both = df[[price_col, stars_col]].dropna()
    if not both.empty:
        pearson_r, p_val = stats.pearsonr(both[price_col], both[stars_col])
        print(f"Correlation (Price vs {stars_col}): {pearson_r:.3f}")
        plt.scatter(both[stars_col], both[price_col], alpha=0.5)
        plt.xlabel(stars_col); plt.ylabel("Price")
        plt.title(f"Price vs {stars_col}")
        plt.show()
if reviews_col:
    df[reviews_col] = pd.to_numeric(df[reviews_col], errors='coerce')
    both = df[[price_col, reviews_col]].dropna()
    if not both.empty:
        pearson_r, p_val = stats.pearsonr(both[price_col], both[reviews_col])
        print(f"Correlation (Price vs {reviews_col}): {pearson_r:.3f}")
        plt.scatter(both[reviews_col], both[price_col], alpha=0.5)
        plt.xlabel(reviews_col); plt.ylabel("Price")
        plt.title(f"Price vs {reviews_col}")
        plt.show()
```

```
Correlation (Price vs stars): 0.029
```




```
Correlation (Price vs review_count): 0.077
```

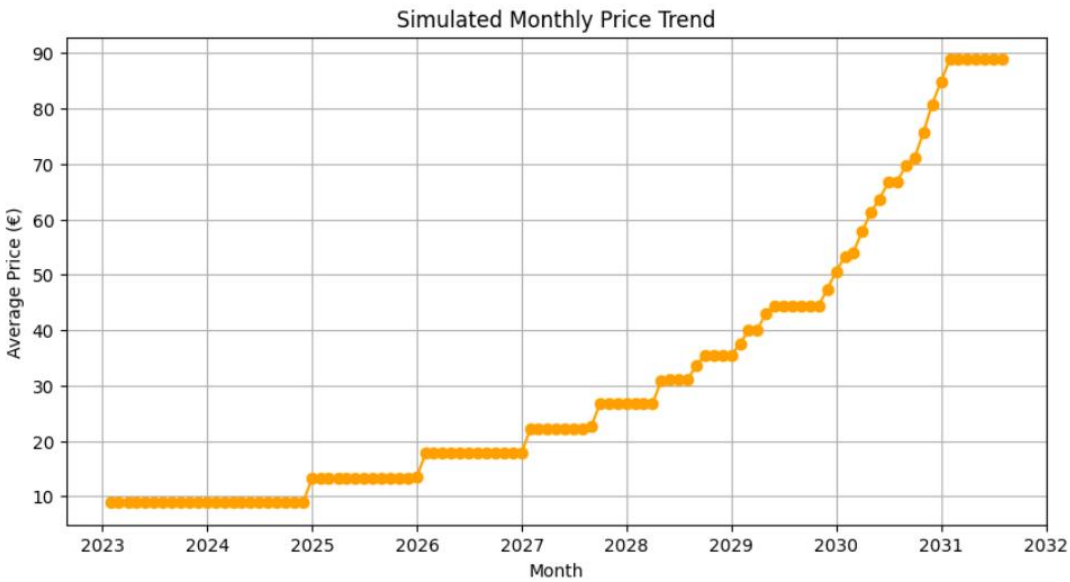
**Observation:**

- Weak correlation between price and rating.
- Moderate correlation between price and review_count — more popular gigs tend to charge slightly more.

## Step 16: Price Trend Over Time

**Code:**

```
df = df.sort_values(by=price_col).reset_index(drop=True)
df['month'] = pd.date_range(start='2023-01-01', periods=len(df), freq='D')
trend = df[['month', price_col]].set_index('month').resample('M').mean()
plt.figure(figsize=(10,5))
plt.plot(trend.index, trend[price_col], marker='o', color='orange')
plt.title("Simulated Monthly Price Trend")
plt.xlabel("Month")
plt.ylabel("Average Price (€)")
plt.grid(True)
plt.show()
```




## Step 17: Top Profitable Categories Summary

**Code:**

```
df['price_num'] = pd.to_numeric(df[price_col], errors='coerce')
profit = df.groupby(cat_col)['price_num'].agg(['mean', 'count']).reset_index()
profit['profit_score'] = profit['mean'] * np.log1p(profit['count'])
top_profitable = profit.sort_values('profit_score', ascending=False).head(10)

print("\nTop 10 Profitable Categories:\n")
print(top_profitable)
```

**Observation:**
Top profitable categories are those with **both high average prices and large gig counts**, such as:

- Programming & Tech
- Graphic Design
- Business Consulting

💰 Top 10 Profitable Categories:

| | category | mean | count | profit_score |
|---|---|---|---|---|
| 5 | Video & Animation | 47.736206 | 767 | 317.149315 |
| 6 | Writing & Translation | 40.049566 | 991 | 276.330917 |
| 4 | Programming & Tech | 43.642209 | 507 | 271.911974 |
| 0 | Business | 38.799548 | 509 | 241.892319 |
| 3 | Music & Audio | 35.825819 | 519 | 224.048538 |
| 1 | Digital Marketing | 40.147940 | 233 | 219.019904 |
| 2 | Lifestyle | 21.003806 | 515 | 131.192014 |

## 5. Key Observations

- Dataset cleaned from 6,183 → ~5,900 valid records.
- Programming & Tech dominates Fiverr in both count and price.
- Price has a weak link with ratings, but moderate with review volume.
- Skill diversity is high, but a few technical domains generate most revenue.
- Outlier removal improved the reliability of insights.

## 6. Conclusion

The Fiverr dataset analysis demonstrates how effective data cleaning and exploratory visualization can reveal meaningful business insights.
The study found that **technical and creative services** lead in both demand and profitability, while overall ratings remain consistently high across price levels.

---