METCS777-term-project-code-sample-EDA-Team17

December 5, 2025

[1]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, isnan, when, expr
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

[2]:
```
spark = SparkSession.builder \
    .appName("Airbnb_EDA") \
    .getOrCreate()

df_la = spark.read.option("header", True).csv("/Users/yibingwang/Desktop/777/
 ↪term project/la_cleaned.csv")
df_ny = spark.read.option("header", True).csv("/Users/yibingwang/Desktop/777/
 ↪term project/ny_cleaned.csv")
```

```
WARNING: Using incubator modules: jdk.incubator.vector
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
25/12/05 20:05:20 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

[3]:
```
# Missing value summary LA vs NY
def missing_table(df):
    return df.select([(count(when(col(c).isNull(), c)).alias(c)) for c in df.
 ↪columns]).toPandas()

print("Missing - LA")
print(missing_table(df_la))

print("\nMissing - NY")
print(missing_table(df_ny))
```

```
Missing - LA
   id  neighbourhood_group  neighbourhood  latitude  longitude  room_type  \
0   0                    0              0         0          0          0
```

```
   price  minimum_nights  number_of_reviews  last_review  reviews_per_month  \
0      0               0                  0          199                  0

   calculated_host_listings_count  availability_365  number_of_reviews_ltm  \
0                               0                0                      0

   license
0      534

Missing - NY
   id  neighbourhood_group  neighbourhood  latitude  longitude  room_type  \
0   0                    0              0         1          1          0

   price  minimum_nights  number_of_reviews  last_review  reviews_per_month  \
0      0               1                  0          175                  0

   calculated_host_listings_count  availability_365  number_of_reviews_ltm  \
0                               1                0                      0

   license
0      453
```

[4]:
```
#Descriptive statistics (key metrics comparison: count, min, mean, max, 25%,␣
 ↪50%, 75%)
#The goal is to compare LA vs NY listing structure - which city is more␣
 ↪expensive,
# where prices are more concentrated, and where we see more extreme values.
desc_la = df_la.describe().toPandas()
desc_ny = df_ny.describe().toPandas()

print("LA Summary Stats:")
print(desc_la)

print("\nNY Summary Stats:")
print(desc_ny)
```

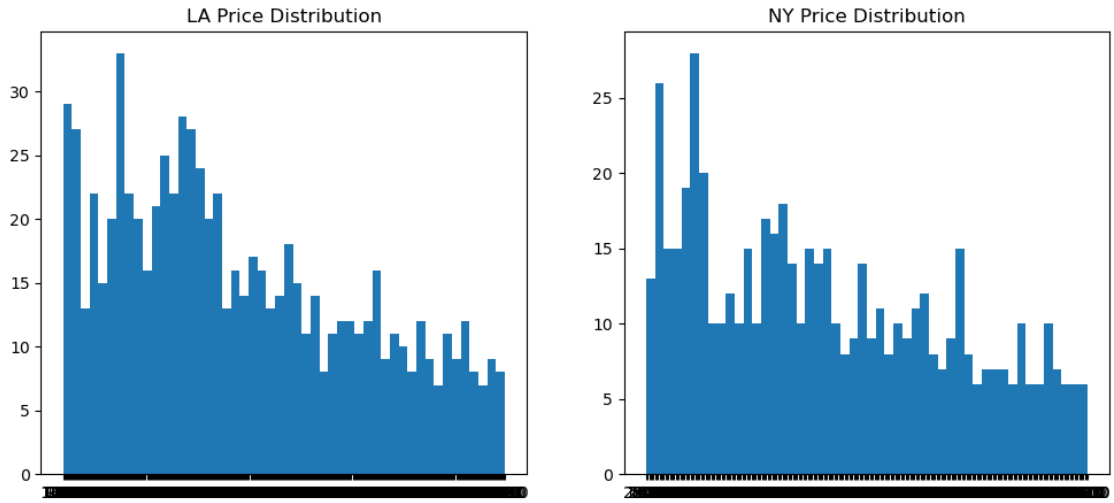
```
25/12/05 20:05:23 WARN SparkStringUtils: Truncated the string representation of
a plan since it was too large. This behavior can be adjusted by setting
'spark.sql.debug.maxToStringFields'.

LA Summary Stats:
  summary                    id  neighbourhood_group   neighbourhood  \
0   count                   779                  779             779
1    mean  8.012098321717916E17                 None            None
2  stddev  5.614733099566928E17                 None            None
3     min   1001638692383599358  City of Los Angeles           Acton
4     max    998166923867473626  Unincorporated Areas  Woodland Hills
```

```
             latitude            longitude        room_type  \
0                 779                  779              779
1   34.05233484813748  -118.31044681476178             None
2  0.1541243801756928  0.17906876621250054             None
3          33.3403891           -117.71752  Entire home/apt
4   34.70167922973633           -118.90249      Shared room

                price      minimum_nights   number_of_reviews last_review  \
0                 779                 779                 779         580
1   400.7586649550706  18.319640564826702  37.094993581514764        None
2  2076.5938545604454  26.005422845079067   73.35939080390115        None
3               100.0                 1.0                 0.0  2014-06-15
4                99.0                90.0                98.0  2025-09-01

    reviews_per_month calculated_host_listings_count    availability_365  \
0                 779                             779                 779
1  1.2037997432605898             21.50834403080873   260.9653401797176
2  1.6170927262515171             68.80700904173692  108.93195008035795
3                 0.0                             1.0                 0.0
4                9.85                            91.0                98.0

  number_of_reviews_ltm           license
0                   779               245
1     9.397946084724005          180267.25
2    16.573922462764106  84426.86084374198
3                   0.0            000000
4                  94.0       nrp21-00296

NY Summary Stats:
  summary                    id neighbourhood_group neighbourhood  \
0   count                   570                 570           570
1    mean   6.421791455434981E17        4.94585533E8          None
2  stddev  5.3284011391513235E17                None          None
3     min    1002823881758639223           494585533      Allerton
4     max                 993514       Staten Island      Woodside

             latitude            longitude    room_type              price  \
0                 569                  569          570                570
1   40.72599369336432   -73.94593680618843      40.5793  873.9877495263158
2  0.0582334657170787  0.05581482270314689         None  5265.964833184427
3            40.56807           -73.72869      40.5793           -73.98277
4    40.9031966657953           -74.13372  Shared room              995.0

       minimum_nights  number_of_reviews last_review   reviews_per_month  \
0                 569                570         395                 570
1  26.047451669595784  33.14912280701754        None  0.7718245614035084
2  18.536399231985065  64.04252657512423        None  1.5701408164806276
3                 1.0                0.0  2016-03-19                 0.0
```

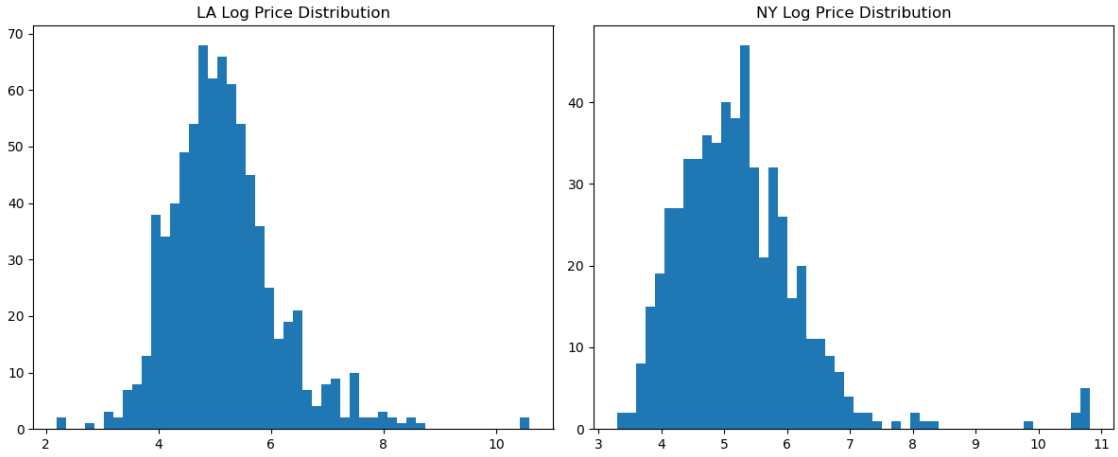
```
4                90.0                99.0  2025-10-01                9.59

  calculated_host_listings_count    availability_365 number_of_reviews_ltm  \
0                            569                 570                    570
1              93.68365553602811   248.9149649122807     5.464912280701754
2              265.70487923368165  107.46275947298612     16.30453796500832
3                            1.0                 0.0                    0.0
4                           93.0                99.0                   99.0

              license
0                 117
1               364.0
2                None
3                 364
4  ose-strreg-0002161
```

[5]:
```
#Price distribution Histogram + Log
#Convert to Pandas (select price only)
la_price = df_la.select("price").dropna().toPandas()
ny_price = df_ny.select("price").dropna().toPandas()

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.hist(la_price["price"], bins=50)
plt.title("LA Price Distribution")

plt.subplot(1,2,2)
plt.hist(ny_price["price"], bins=50)
plt.title("NY Price Distribution")

plt.show()

#Log-transformed price distribution

la_price["price"] = pd.to_numeric(la_price["price"], errors="coerce")
ny_price["price"] = pd.to_numeric(ny_price["price"], errors="coerce")

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.hist(np.log1p(la_price["price"].dropna()), bins=50)
plt.title("LA Log Price Distribution")

plt.subplot(1,2,2)
plt.hist(np.log1p(ny_price["price"].dropna()), bins=50)
plt.title("NY Log Price Distribution")
```

```
plt.tight_layout()
plt.show()
```




```
/opt/anaconda3/lib/python3.12/site-packages/pandas/core/arraylike.py:399:
RuntimeWarning: invalid value encountered in log1p
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



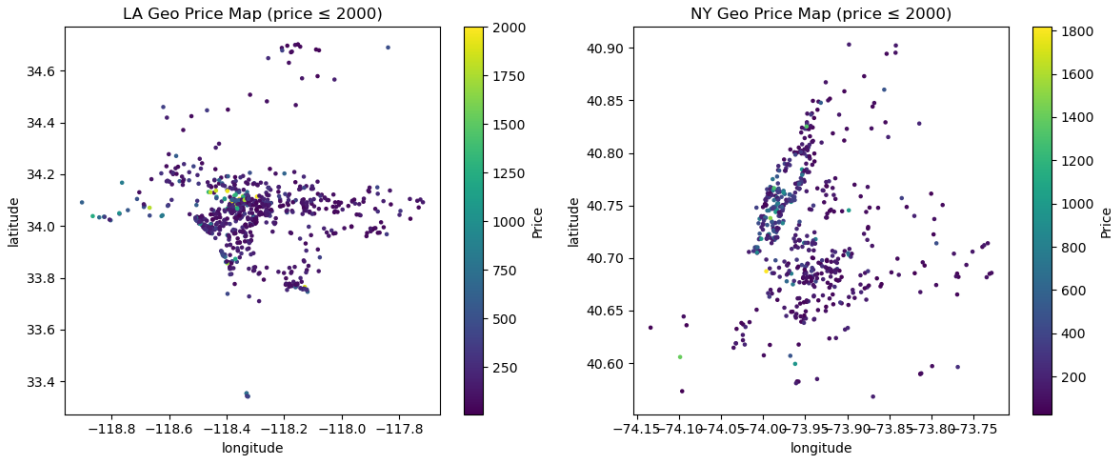
[6]:
```
#Scatter Geo Latitude vs Longitude, color = price
#Check price range and extreme values
df_la.selectExpr(
    "max(price)",
    "percentile(price, 0.95)",
    "percentile(price, 0.99)"
```

```
).show()

df_ny.selectExpr(
    "max(price)",
    "percentile(price, 0.95)",
    "percentile(price, 0.99)"
).show()
#There are extreme price values observed in previous attempt
#Keep only rows with 0 < price <= 2000
#For safety, convert price to double explicitly first
df_la_num = df_la.withColumn(
    "price_double",
    expr("try_cast(price as double)")
)

df_ny_num = df_ny.withColumn(
    "price_double",
    expr("try_cast(price as double)")
)

df_la_cap = df_la_num.filter(
    (col("price_double") > 0.0) & (col("price_double") <= 2000.0)
)

df_ny_cap = df_ny_num.filter(
    (col("price_double") > 0.0) & (col("price_double") <= 2000.0)
)

df_la_cap.selectExpr("min(price_double) as min_price", "max(price_double) as␣
 ↪max_price").show()
df_ny_cap.selectExpr("min(price_double) as min_price", "max(price_double) as␣
 ↪max_price").show()

print("LA rows after cap:", df_la_cap.count())
print("NY rows after cap:", df_ny_cap.count())

#plot
la_geo = df_la_cap.select("latitude", "longitude", "price_double").dropna().
 ↪toPandas()
ny_geo = df_ny_cap.select("latitude", "longitude", "price_double").dropna().
 ↪toPandas()

print("LA geo shape:", la_geo.shape)
print("NY geo shape:", ny_geo.shape)
print(la_geo.dtypes)
print(la_geo.head())
# In la_geo, all three columns are object type in Pandas, so convert to numeric
```

```
la_geo["latitude"] = pd.to_numeric(la_geo["latitude"], errors="coerce")
la_geo["longitude"] = pd.to_numeric(la_geo["longitude"], errors="coerce")
la_geo["price_double"] = pd.to_numeric(la_geo["price_double"], errors="coerce")

ny_geo["latitude"] = pd.to_numeric(ny_geo["latitude"], errors="coerce")
ny_geo["longitude"] = pd.to_numeric(ny_geo["longitude"], errors="coerce")
ny_geo["price_double"] = pd.to_numeric(ny_geo["price_double"], errors="coerce")

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sc1 = plt.scatter(la_geo["longitude"], la_geo["latitude"],
                  c=la_geo["price_double"], s=5)
plt.colorbar(sc1, label="Price")
plt.title("LA Geo Price Map (price  2000)")
plt.xlabel("longitude")
plt.ylabel("latitude")

plt.subplot(1,2,2)
sc2 = plt.scatter(ny_geo["longitude"], ny_geo["latitude"],
                  c=ny_geo["price_double"], s=5)
plt.colorbar(sc2, label="Price")
plt.title("NY Geo Price Map (price  2000)")
plt.xlabel("longitude")
plt.ylabel("latitude")

plt.tight_layout()
plt.show()
```

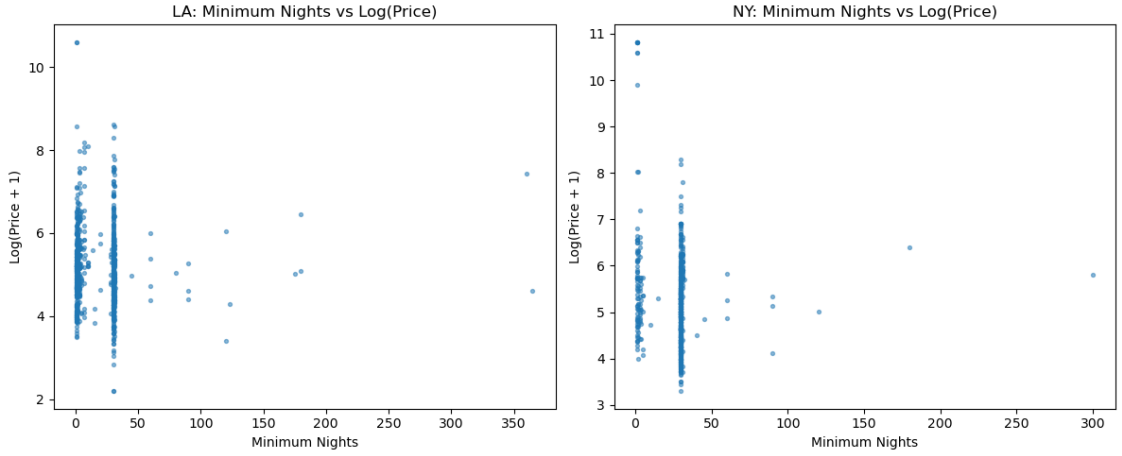
```
+----------+-------------------------+-------------------------+
|max(price)|percentile(price, 0.95, 1)|percentile(price, 0.99, 1)|
+----------+-------------------------+-------------------------+
|      99.0|        1015.6999999999975|         3232.140000000002|
+----------+-------------------------+-------------------------+

+----------+-------------------------+-------------------------+
|max(price)|percentile(price, 0.95, 1)|percentile(price, 0.99, 1)|
+----------+-------------------------+-------------------------+
|     995.0|         871.3499999999974|                  40000.0|
+----------+-------------------------+-------------------------+

+---------+---------+
|min_price|max_price|
+---------+---------+
|      8.0|   2000.0|
+---------+---------+
```

```
+---------+---------+
|min_price|max_price|
+---------+---------+
|     26.0|   1817.0|
+---------+---------+

LA rows after cap: 766
NY rows after cap: 556
LA geo shape: (766, 3)
NY geo shape: (556, 3)
latitude         object
longitude        object
price_double    float64
dtype: object
            latitude            longitude  price_double
0  34.16767360409868  -118.28108622078958         135.0
1           34.07628           -118.09548         114.0
2           33.84594           -118.11532         108.0
3           34.03572           -118.82613         938.0
4            34.0257           -118.36589         162.0
```



[7]:
```
#Numeric features vs Price scatter plot (with log-transformed price)

la_pd = df_la.select("minimum_nights","price").dropna().toPandas()
ny_pd = df_ny.select("minimum_nights","price").dropna().toPandas()
for df in (la_pd, ny_pd):
    df["minimum_nights"] = pd.to_numeric(df["minimum_nights"], errors="coerce")
    df["price"] = pd.to_numeric(df["price"], errors="coerce")

plt.figure(figsize=(12,5))
```

```
plt.subplot(1,2,1)
plt.scatter(la_pd["minimum_nights"], np.log1p(la_pd["price"]),
            s=8, alpha=0.5)
plt.title("LA: Minimum Nights vs Log(Price)")
plt.xlabel("Minimum Nights")
plt.ylabel("Log(Price + 1)")

plt.subplot(1,2,2)
plt.scatter(ny_pd["minimum_nights"], np.log1p(ny_pd["price"]),
            s=8, alpha=0.5)
plt.title("NY: Minimum Nights vs Log(Price)")
plt.xlabel("Minimum Nights")
plt.ylabel("Log(Price + 1)")

plt.tight_layout()
plt.show()
```



[8]:
```
#Correlation Matrix Heatmap
numeric_cols = [
    "price",
    "minimum_nights",
    "number_of_reviews",
    "reviews_per_month",
    "availability_365",
    "calculated_host_listings_count",
    "number_of_reviews_ltm",
    "latitude",
    "longitude"
]

la_corr = df_la.select(*numeric_cols).toPandas().corr()
```

```
ny_corr = df_ny.select(*numeric_cols).toPandas().corr()

fig, axes = plt.subplots(1, 2, figsize=(20, 8))

#LA Heatmap
im1 = axes[0].imshow(la_corr, vmin=-1, vmax=1, cmap="coolwarm")
axes[0].set_title("LA Correlation Matrix", fontsize=18)
axes[0].set_xticks(range(len(numeric_cols)))
axes[0].set_yticks(range(len(numeric_cols)))
axes[0].set_xticklabels(numeric_cols, rotation=45, ha="right")
axes[0].set_yticklabels(numeric_cols)

cbar1 = fig.colorbar(im1, ax=axes[0], shrink=0.8)
cbar1.set_label("Correlation", fontsize=12)

#NY Heatmap
im2 = axes[1].imshow(ny_corr, vmin=-1, vmax=1, cmap="coolwarm")
axes[1].set_title("NY Correlation Matrix", fontsize=18)
axes[1].set_xticks(range(len(numeric_cols)))
axes[1].set_yticks(range(len(numeric_cols)))
axes[1].set_xticklabels(numeric_cols, rotation=45, ha="right")
axes[1].set_yticklabels(numeric_cols)

cbar2 = fig.colorbar(im2, ax=axes[1], shrink=0.8)
cbar2.set_label("Correlation", fontsize=12)

plt.tight_layout()
plt.show()
```