FINAL PROJECT 1

LINEAR REGRESSION

KELAS PYTN-10

Kelompok 1 :

1. ELSA INDRIANI
2. FELIS TIGRIS HAFIZULLOH
3. HANI NAFISAH AMALIYA

Objektif

Final Project 1 ini dibuat guna mengevaluasi konsep Regression sebagai berikut :

- Mampu memahami konsep regression dengan Linear Regression
- Mampu mempersiapkan data untuk digunakan dalam model Linear Regression
- Mampu mengimplementasikan Linear Regression untuk membuat prediksi

Project Overview

Database ini memiliki 57 atribut, tetapi yang paling relevan ada 10 atribut dari semuanya.

Final Project 1 ini dibuat guna mengevaluasi konsep Regression sebagai berikut :

- Mampu memahami konsep regression dengan Linear Regression
- Mampu mempersiapkan data untuk digunakan dalam model Linear Regression
- Mampu mengimplementasikan Linear Regression untuk membuat prediksi

Attribute Information :

1. id
2. timestamp
3. hour
4. day
5. month
6. datetime
7. timezone
8. source : destinasi awal
9. destination : destinasi akhir
10. cab_type : tipe transportasi (uber / lyft)
11. … dan lainnya

In [1]:
```
# Import Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix, accu
racy_score
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

In [2]:
```
# Membaca File
df = pd.read_csv('./dataset/rideshare_kaggle.csv')
df
```

Out[2]:

	id	timestamp	hour	day	month	datetime	timezon
0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	1.544953e+09	9	16	12	2018-12-16 09:30:07	America/New_Yoı
1	4bd23055-6827-41c6-b23b-3c491f24e74d	1.543284e+09	2	27	11	2018-11-27 02:00:23	America/New_Yoı
2	981a3613-77af-4620-a42a-0c0866077d1e	1.543367e+09	1	28	11	2018-11-28 01:00:22	America/New_Yoı
3	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	1.543554e+09	4	30	11	2018-11-30 04:53:02	America/New_Yoı
4	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	1.543463e+09	3	29	11	2018-11-29 03:49:20	America/New_Yoı
...	...	...	...	...	...	...	.
693066	616d3611-1820-450a-9845-a9ff304a4842	1.543708e+09	23	1	12	2018-12-01 23:53:05	America/New_Yoı
693067	633a3fc3-1f86-4b9e-9d48-2b7132112341	1.543708e+09	23	1	12	2018-12-01 23:53:05	America/New_Yoı
693068	64d451d0-639f-47a4-9b7c-6fd92fbd264f	1.543708e+09	23	1	12	2018-12-01 23:53:05	America/New_Yoı
693069	727e5f07-a96b-4ad1-a2c7-9abc3ad55b4e	1.543708e+09	23	1	12	2018-12-01 23:53:05	America/New_Yoı
693070	e7fdc087-fe86-40a5-a3c3-3b2a8badcbda	1.543708e+09	23	1	12	2018-12-01 23:53:05	America/New_Yoı

693071 rows × 57 columns

In [3]:
```
# Menampilkan jumlah baris dan kolom
df.shape
```

Out[3]: (693071, 57)

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 693071 entries, 0 to 693070
Data columns (total 57 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   id                           693071 non-null  object
 1   timestamp                    693071 non-null  float64
 2   hour                         693071 non-null  int64
 3   day                          693071 non-null  int64
 4   month                        693071 non-null  int64
 5   datetime                     693071 non-null  object
 6   timezone                     693071 non-null  object
 7   source                       693071 non-null  object
 8   destination                  693071 non-null  object
 9   cab_type                     693071 non-null  object
 10  product_id                   693071 non-null  object
 11  name                         693071 non-null  object
 12  price                        637976 non-null  float64
 13  distance                     693071 non-null  float64
 14  surge_multiplier             693071 non-null  float64
 15  latitude                     693071 non-null  float64
 16  longitude                    693071 non-null  float64
 17  temperature                  693071 non-null  float64
 18  apparentTemperature          693071 non-null  float64
 19  short_summary                693071 non-null  object
 20  long_summary                 693071 non-null  object
 21  precipIntensity              693071 non-null  float64
 22  precipProbability            693071 non-null  float64
 23  humidity                     693071 non-null  float64
 24  windSpeed                    693071 non-null  float64
 25  windGust                     693071 non-null  float64
 26  windGustTime                 693071 non-null  int64
 27  visibility                   693071 non-null  float64
 28  temperatureHigh              693071 non-null  float64
 29  temperatureHighTime          693071 non-null  int64
 30  temperatureLow               693071 non-null  float64
 31  temperatureLowTime           693071 non-null  int64
 32  apparentTemperatureHigh      693071 non-null  float64
 33  apparentTemperatureHighTime  693071 non-null  int64
 34  apparentTemperatureLow       693071 non-null  float64
 35  apparentTemperatureLowTime   693071 non-null  int64
 36  icon                         693071 non-null  object
 37  dewPoint                     693071 non-null  float64
 38  pressure                     693071 non-null  float64
 39  windBearing                  693071 non-null  int64
 40  cloudCover                   693071 non-null  float64
 41  uvIndex                      693071 non-null  int64
 42  visibility.1                 693071 non-null  float64
 43  ozone                        693071 non-null  float64
 44  sunriseTime                  693071 non-null  int64
 45  sunsetTime                   693071 non-null  int64
 46  moonPhase                    693071 non-null  float64
 47  precipIntensityMax           693071 non-null  float64
 48  uvIndexTime                  693071 non-null  int64
 49  temperatureMin               693071 non-null  float64
 50  temperatureMinTime           693071 non-null  int64
 51  temperatureMax               693071 non-null  float64
 52  temperatureMaxTime           693071 non-null  int64
 53  apparentTemperatureMin       693071 non-null  float64
 54  apparentTemperatureMinTime   693071 non-null  int64
```

```
 55  apparentTemperatureMax       693071 non-null  float64
 56  apparentTemperatureMaxTime   693071 non-null  int64
dtypes: float64(29), int64(17), object(11)
memory usage: 301.4+ MB
```

In [5]:

```
df.isnull().sum()
```

Out[5]:
```
id                             0
timestamp                      0
hour                           0
day                            0
month                          0
datetime                       0
timezone                       0
source                         0
destination                    0
cab_type                       0
product_id                     0
name                           0
price                      55095
distance                       0
surge_multiplier               0
latitude                       0
longitude                      0
temperature                    0
apparentTemperature            0
short_summary                  0
long_summary                   0
precipIntensity                0
precipProbability              0
humidity                       0
windSpeed                      0
windGust                       0
windGustTime                   0
visibility                     0
temperatureHigh                0
temperatureHighTime            0
temperatureLow                 0
temperatureLowTime             0
apparentTemperatureHigh        0
apparentTemperatureHighTime    0
apparentTemperatureLow         0
apparentTemperatureLowTime     0
icon                           0
dewPoint                       0
pressure                       0
windBearing                    0
cloudCover                     0
uvIndex                        0
visibility.1                   0
ozone                          0
sunriseTime                    0
sunsetTime                     0
moonPhase                      0
precipIntensityMax             0
uvIndexTime                    0
temperatureMin                 0
temperatureMinTime             0
temperatureMax                 0
temperatureMaxTime             0
apparentTemperatureMin         0
apparentTemperatureMinTime     0
apparentTemperatureMax         0
apparentTemperatureMaxTime     0
dtype: int64
```

Data Cleaning

Kolom price memiliki nilai null sebesar 55095. Pada tahap ini tidak akan mengisi missing value pada fitur price dikarenakan fitur price merupakan variable dependent pada projek ini,jika memaksa mengisi missing value pada fitur price akan mengakibatkan lebih banyak nilai yang error dan akurasi yang kurang. Jadi hapus semua record dimana kolom price memiliki nilai null.

In [6]:
```
df = df.dropna(subset=['price']).reset_index()
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
index                          0
id                             0
timestamp                      0
hour                           0
day                            0
month                          0
datetime                       0
timezone                       0
source                         0
destination                    0
cab_type                       0
product_id                     0
name                           0
price                          0
distance                       0
surge_multiplier               0
latitude                       0
longitude                      0
temperature                    0
apparentTemperature            0
short_summary                  0
long_summary                   0
precipIntensity                0
precipProbability              0
humidity                       0
windSpeed                      0
windGust                       0
windGustTime                   0
visibility                     0
temperatureHigh                0
temperatureHighTime            0
temperatureLow                 0
temperatureLowTime             0
apparentTemperatureHigh        0
apparentTemperatureHighTime    0
apparentTemperatureLow         0
apparentTemperatureLowTime     0
icon                           0
dewPoint                       0
pressure                       0
windBearing                    0
cloudCover                     0
uvIndex                        0
visibility.1                   0
ozone                          0
sunriseTime                    0
sunsetTime                     0
moonPhase                      0
precipIntensityMax             0
uvIndexTime                    0
temperatureMin                 0
temperatureMinTime             0
temperatureMax                 0
temperatureMaxTime             0
apparentTemperatureMin         0
apparentTemperatureMinTime     0
apparentTemperatureMax         0
apparentTemperatureMaxTime     0
dtype: int64
```

In [8]:
```
# Hapus fitur yang tidak memiliki dependency terhadap predictand (price)
# Dikarenakan sudah ada atribut jarak dan parameter waktu (hours, day, mo
nth)
# Hapus fitur latitude, longitude dan datetime dari dataframe

df = df.drop(['id','timestamp','datetime','long_summary','apparentTempera
tureHighTime','apparentTemperatureLowTime',
             'windGustTime','sunriseTime','sunsetTime','uvIndexTime','tem
peratureMinTime','temperatureMaxTime',
             'apparentTemperatureMinTime','temperatureLowTime','apparentT
emperatureMaxTime','latitude','longitude'],
            axis=1)
print(df.shape)
df.head()
```

(637976, 41)

Out[8]:

	index	hour	day	month	timezone	source	destination	cab_type	product_
0	0	9	16	12	America/New_York	Haymarket Square	North Station	Lyft	lyft_lir
1	1	2	27	11	America/New_York	Haymarket Square	North Station	Lyft	lyft_premi
2	2	1	28	11	America/New_York	Haymarket Square	North Station	Lyft	ly
3	3	4	30	11	America/New_York	Haymarket Square	North Station	Lyft	lyft_luxsu
4	4	3	29	11	America/New_York	Haymarket Square	North Station	Lyft	lyft_plu

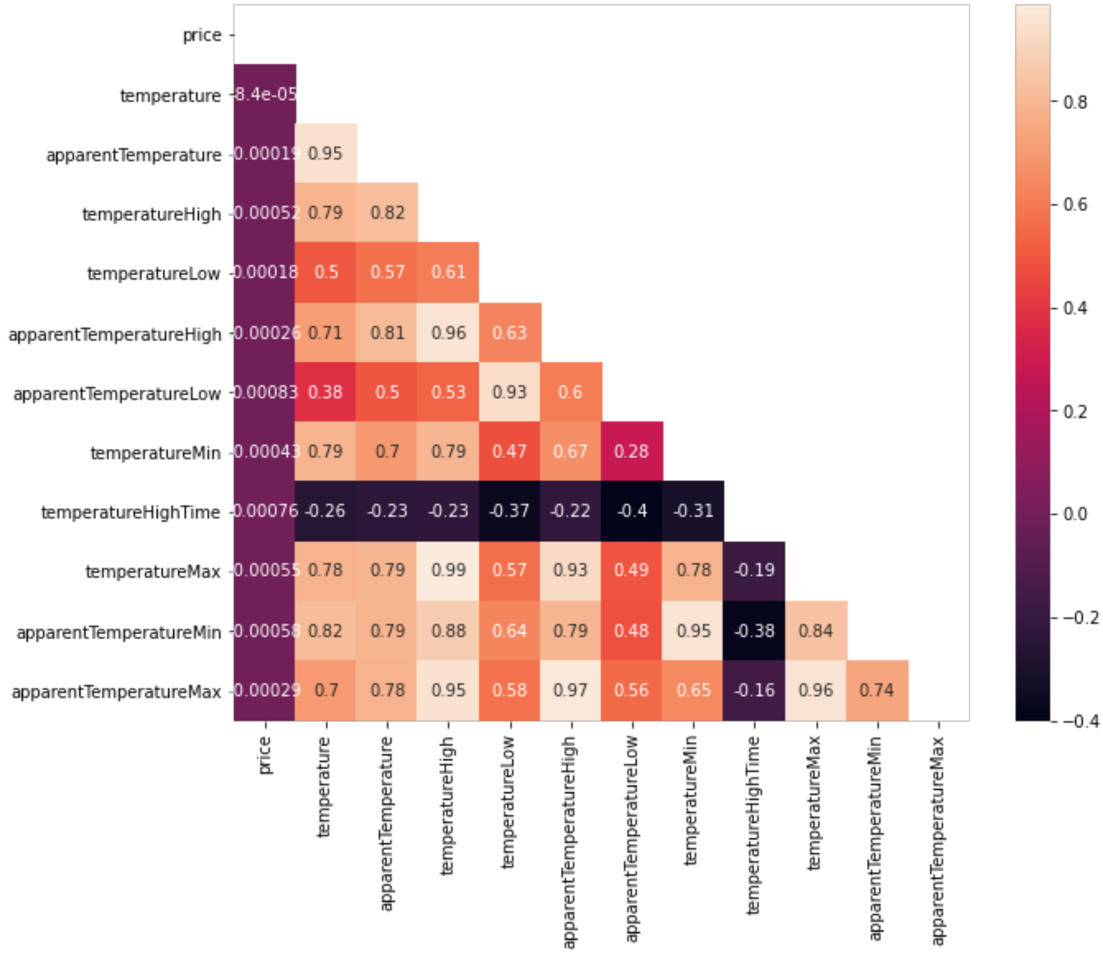
5 rows × 41 columns

In [9]:
```
# Cek korelasi antara fitur price dan fitur yang memiliki relasi dengan t
emperature
new_df = df[['price','temperature','apparentTemperature','temperatureHigh
','temperatureLow','apparentTemperatureHigh',
            'apparentTemperatureLow','temperatureMin','temperatureHighTim
e','temperatureMax','apparentTemperatureMin',
            'apparentTemperatureMax']]
new_df.head()
```

Out[9]:

	price	temperature	apparentTemperature	temperatureHigh	temperatureLow	apparentTen
0	5.0	42.34	37.12	43.68	34.19	
1	11.0	43.58	37.35	47.30	42.10	
2	7.0	38.33	32.93	47.55	33.10	
3	26.0	34.38	29.63	45.03	28.90	
4	9.0	37.44	30.88	42.18	36.71	

In [10]:
```python
# Gunakan Heatmap Plot dengan correlation untuk melihat rate korelasi pad
a dataframe baru
plt.figure(figsize=(10,8))
sns.heatmap(new_df.corr(), annot=True, mask=np.triu(new_df.corr()));
```



In [11]:
```
# Hapus semua fitur yang berhubungan dengan temperatur dari dataframe
# Dikarenakan mereka memiliki korelasi yang lemah terhadap predictand (pr
ice)
new_df = df[['temperature','apparentTemperature','temperatureHigh','tempe
ratureLow','apparentTemperatureHigh',
            'apparentTemperatureLow','temperatureMin','temperatureHighTim
e','temperatureMax','apparentTemperatureMin',
            'apparentTemperatureMax']]
df = df.drop(new_df.columns, axis=1)
print(df.shape)
df.head()
```

```
(637976, 30)
```

Out[11]:

	index	hour	day	month	timezone	source	destination	cab_type	product_
0	0	9	16	12	America/New_York	Haymarket Square	North Station	Lyft	lyft_lir
1	1	2	27	11	America/New_York	Haymarket Square	North Station	Lyft	lyft_premi
2	2	1	28	11	America/New_York	Haymarket Square	North Station	Lyft	ly
3	3	4	30	11	America/New_York	Haymarket Square	North Station	Lyft	lyft_luxsu
4	4	3	29	11	America/New_York	Haymarket Square	North Station	Lyft	lyft_plu

5 rows × 30 columns

In [12]:
```
# Eksplorasi dan analisis data pada fitur dengan tipe categorical
categorical_cols = df.select_dtypes(include=['object','category']).column
s.tolist()
new_data = df[categorical_cols]
new_data.head()
```

Out[12]:

	timezone	source	destination	cab_type	product_id	name	short_summary
0	America/New_York	Haymarket Square	North Station	Lyft	lyft_line	Shared	Mostly Cloudy
1	America/New_York	Haymarket Square	North Station	Lyft	lyft_premier	Lux	Rain
2	America/New_York	Haymarket Square	North Station	Lyft	lyft	Lyft	Clear
3	America/New_York	Haymarket Square	North Station	Lyft	lyft_luxsuv	Lux Black XL	Clear
4	America/New_York	Haymarket Square	North Station	Lyft	lyft_plus	Lyft XL	Partly Cloudy

In [13]:
```
# Cek nilai unik pada setiap kolom yang bertipe categorical
for col in new_data:
    print(f"{col} :{new_data[col].unique()}")
    print()
```

```
timezone :['America/New_York']

source :['Haymarket Square' 'Back Bay' 'North End' 'North Station' 'Beac
on Hill'
 'Boston University' 'Fenway' 'South Station' 'Theatre District'
 'West End' 'Financial District' 'Northeastern University']

destination :['North Station' 'Northeastern University' 'West End' 'Haym
arket Square'
 'South Station' 'Fenway' 'Theatre District' 'Beacon Hill' 'Back Bay'
 'North End' 'Financial District' 'Boston University']

cab_type :['Lyft' 'Uber']

product_id :['lyft_line' 'lyft_premier' 'lyft' 'lyft_luxsuv' 'lyft_plus'
'lyft_lux'
 '6f72dfc5-27f1-42e8-84db-ccc7a75f6969'
 '6c84fd89-3f11-4782-9b50-97c468b19529'
 '55c66225-fbe7-4fd5-9072-eab1ece5e23e'
 '9a0e7b09-b92b-4c41-9779-2ad22b4d779d'
 '6d318bcc-22a3-4af6-bddd-b409bfce1546'
 '997acbb5-e102-41e1-b155-9df7de0a73f2']

name :['Shared' 'Lux' 'Lyft' 'Lux Black XL' 'Lyft XL' 'Lux Black' 'UberX
L'
 'Black' 'UberX' 'WAV' 'Black SUV' 'UberPool']

short_summary :[' Mostly Cloudy ' ' Rain ' ' Clear ' ' Partly Cloudy ' '
Overcast '
 ' Light Rain ' ' Foggy ' ' Possible Drizzle ' ' Drizzle ']

icon :[' partly-cloudy-night ' ' rain ' ' clear-night ' ' cloudy ' ' fog
'
 ' clear-day ' ' partly-cloudy-day ']
```

In [14]:
```
new_data['product_id'].value_counts()
```

Out[14]:
```
6f72dfc5-27f1-42e8-84db-ccc7a75f6969    55096
9a0e7b09-b92b-4c41-9779-2ad22b4d779d    55096
6d318bcc-22a3-4af6-bddd-b409bfce1546    55096
6c84fd89-3f11-4782-9b50-97c468b19529    55095
55c66225-fbe7-4fd5-9072-eab1ece5e23e    55094
997acbb5-e102-41e1-b155-9df7de0a73f2    55091
lyft_lux                                51235
lyft_premier                            51235
lyft_luxsuv                             51235
lyft_plus                               51235
lyft                                    51235
lyft_line                               51233
Name: product_id, dtype: int64
```

In [15]:
```python
# Hapus timezone dan product_id karena berisi data yang tidak diperlukan
df = df.drop(['timezone','product_id'], axis=1)
df.head()
```

Out[15]:

	index	hour	day	month	source	destination	cab_type	name	price	distance	...
0	0	9	16	12	Haymarket Square	North Station	Lyft	Shared	5.0	0.44	...
1	1	2	27	11	Haymarket Square	North Station	Lyft	Lux	11.0	0.44	...
2	2	1	28	11	Haymarket Square	North Station	Lyft	Lyft	7.0	0.44	...
3	3	4	30	11	Haymarket Square	North Station	Lyft	Lux Black XL	26.0	0.44	...
4	4	3	29	11	Haymarket Square	North Station	Lyft	Lyft XL	9.0	0.44	...

5 rows × 28 columns

In [16]:
```python
# Analisis kolom/fitur yang memiliki tipe numerical
num_cols = df.select_dtypes(include=['int64','float64']).columns.tolist()
new_data = df[num_cols]
new_data.columns
```

Out[16]:
```
Index(['index', 'hour', 'day', 'month', 'price', 'distance',
       'surge_multiplier', 'precipIntensity', 'precipProbability', 'humi
dity',
       'windSpeed', 'windGust', 'visibility', 'dewPoint', 'pressure',
       'windBearing', 'cloudCover', 'uvIndex', 'visibility.1', 'ozone',
       'moonPhase', 'precipIntensityMax'],
      dtype='object')
```

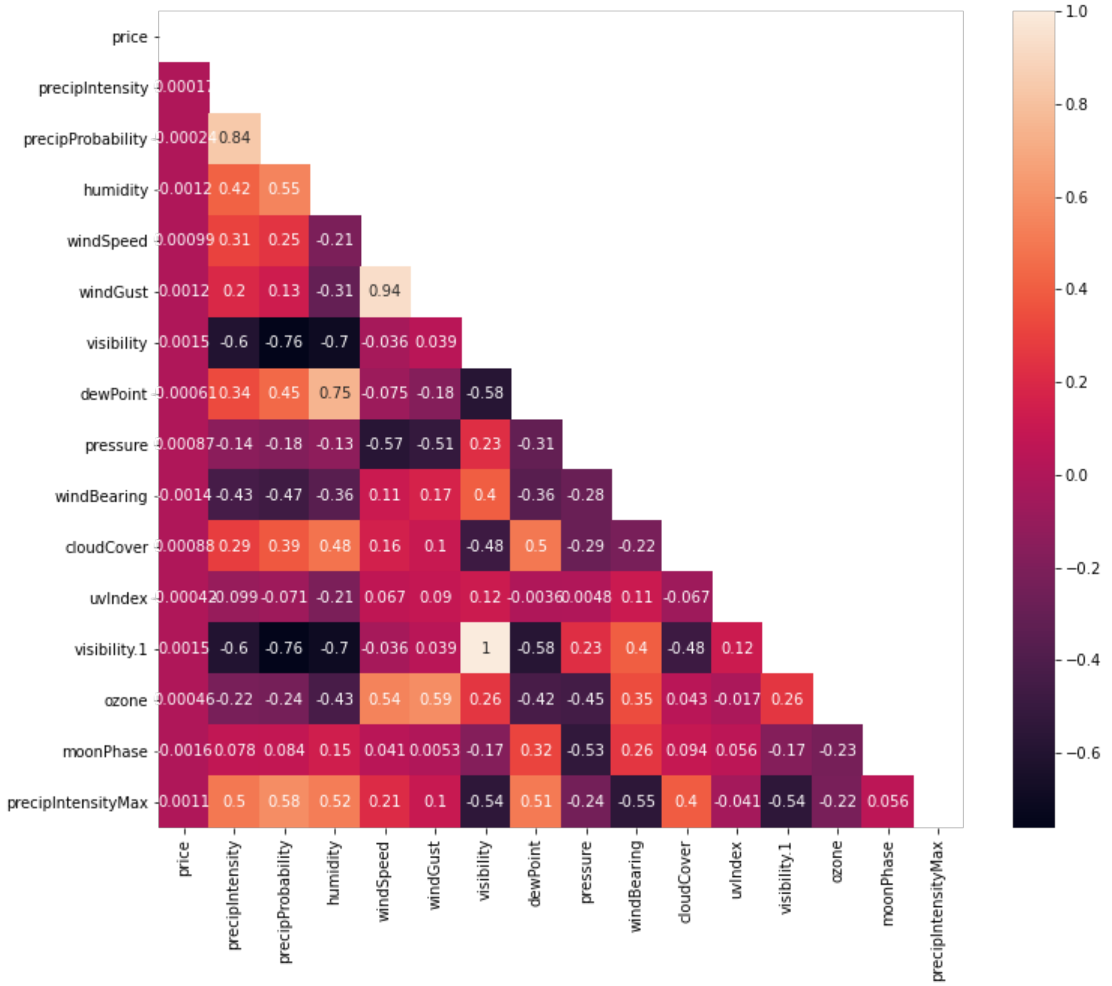
In [17]:
```python
# Analisis dan cek tingkat korelasi antara fitur yang berhubungan dengan
cuaca dan price
climate_cols = ['price','precipIntensity','precipProbability','humidity
','windSpeed','windGust',
               'visibility','dewPoint','pressure','windBearing','cloudCov
er','uvIndex','visibility.1',
               'ozone','moonPhase','precipIntensityMax',]
new_data = df[climate_cols]
new_data.head()
```

Out[17]:

	price	precipIntensity	precipProbability	humidity	windSpeed	windGust	visibility	dewP
0	5.0	0.0000	0.0	0.68	8.66	9.17	10.000	3
1	11.0	0.1299	1.0	0.94	11.98	11.98	4.786	4
2	7.0	0.0000	0.0	0.75	7.33	7.33	10.000	3
3	26.0	0.0000	0.0	0.73	5.28	5.28	10.000	2
4	9.0	0.0000	0.0	0.70	9.14	9.14	10.000	2

In [18]:

```
plt.figure(figsize=(12,10))
sns.heatmap(new_data.corr(), annot=True, mask=np.triu(new_data.corr()));
```



Semua fitur yang berhubungan dengan cuaca memiliki korelasi yang **rendah hampir 0** terhadap kolom price.

In [19]:
```
# Drop climate_cols
climate_cols = ['precipIntensity','precipProbability','humidity','windSpe
ed','windGust',
               'visibility','dewPoint','pressure','windBearing','cloudCov
er','uvIndex','visibility.1',
               'ozone','moonPhase','precipIntensityMax']
df = df.drop(climate_cols, axis=1)
print(df.shape)
df.head()
```

```
(637976, 13)
```

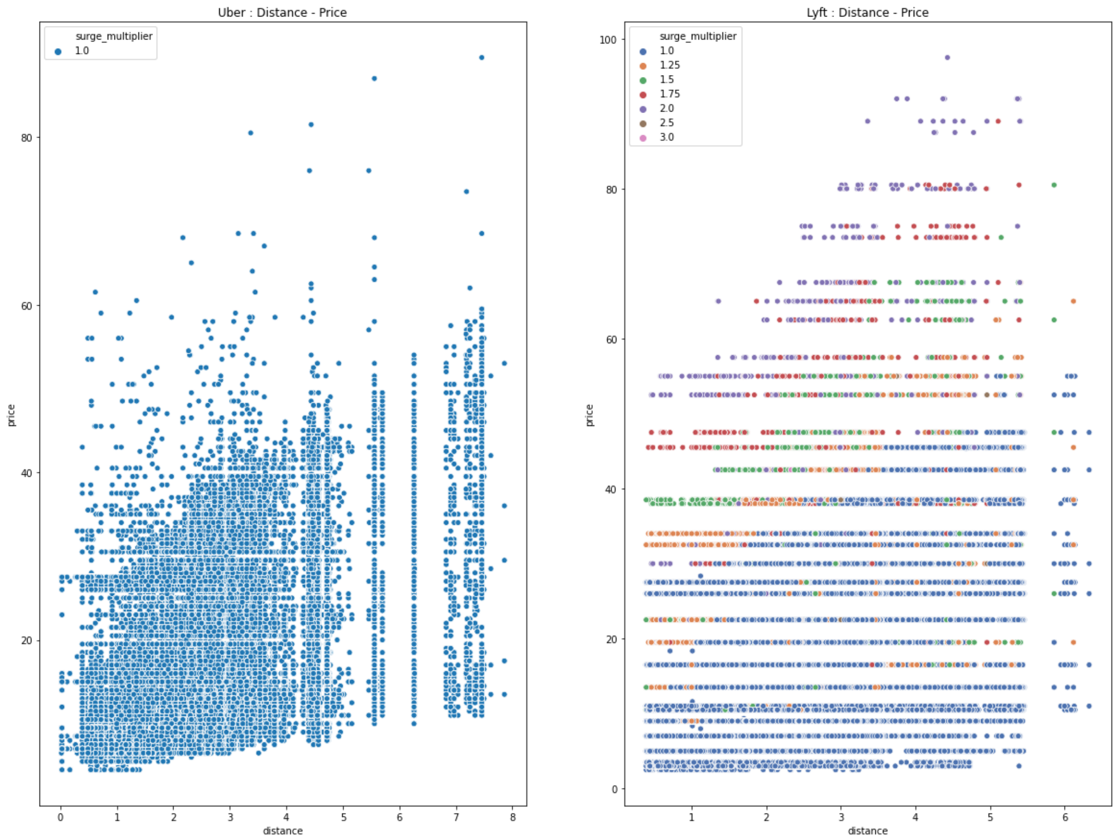
Out[19]:

	index	hour	day	month	source	destination	cab_type	name	price	distance	su
0	0	9	16	12	Haymarket Square	North Station	Lyft	Shared	5.0	0.44	
1	1	2	27	11	Haymarket Square	North Station	Lyft	Lux	11.0	0.44	
2	2	1	28	11	Haymarket Square	North Station	Lyft	Lyft	7.0	0.44	
3	3	4	30	11	Haymarket Square	North Station	Lyft	Lux Black XL	26.0	0.44	
4	4	3	29	11	Haymarket Square	North Station	Lyft	Lyft XL	9.0	0.44	

Exploratory Data Analysis

In [20]:
```
uber_data = df[df['cab_type']=='Uber']
lyft_data = df[df['cab_type']=='Lyft']
```

In [21]:
```
uber_sm_dis_price = uber_data[['distance','surge_multiplier','price']]
lyft_sm_dis_price = lyft_data[['surge_multiplier','distance','price']]

# plotting menggunakan scatter plot
plt.figure(figsize=(20,15))
plt.subplot(121)
sns.scatterplot(data = uber_sm_dis_price, x = 'distance', y='price', hue=
'surge_multiplier').set_title("Uber : Distance - Price");

plt.subplot(122)
sns.scatterplot(data = lyft_sm_dis_price, x = 'distance', y='price', hue=
'surge_multiplier', palette='deep').set_title("Lyft : Distance - Price");
```

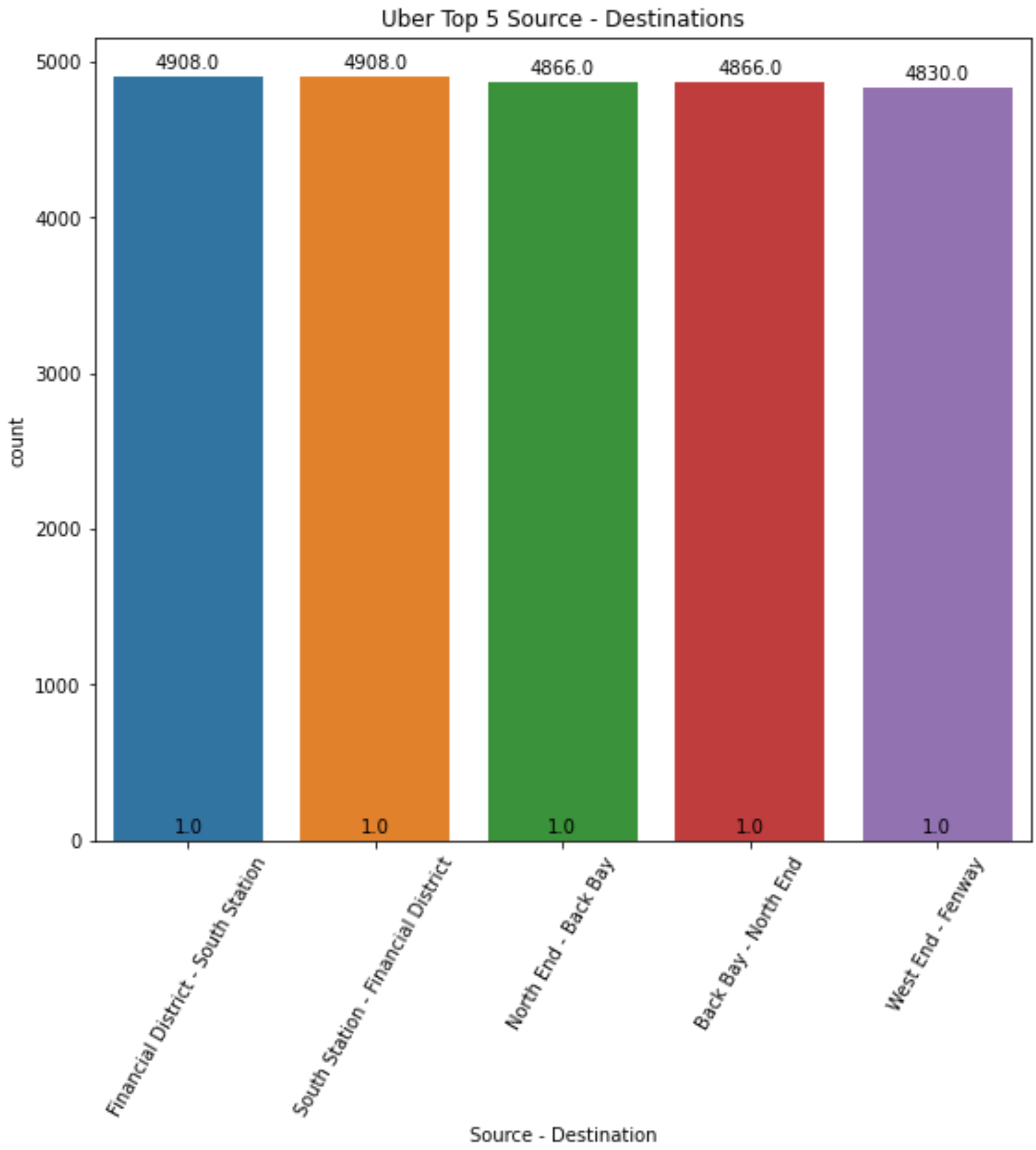


Dari visualisasi untuk data **Uber** diatas dapat dilihat bahwa distance dan price **tidak** memiliki korelasi yang kuat, price tidak bertambah secara linier. Sedangkan pada data **Lyft** ketika distance bertambah, harga ikut bertambah.

Top 5 Destinations

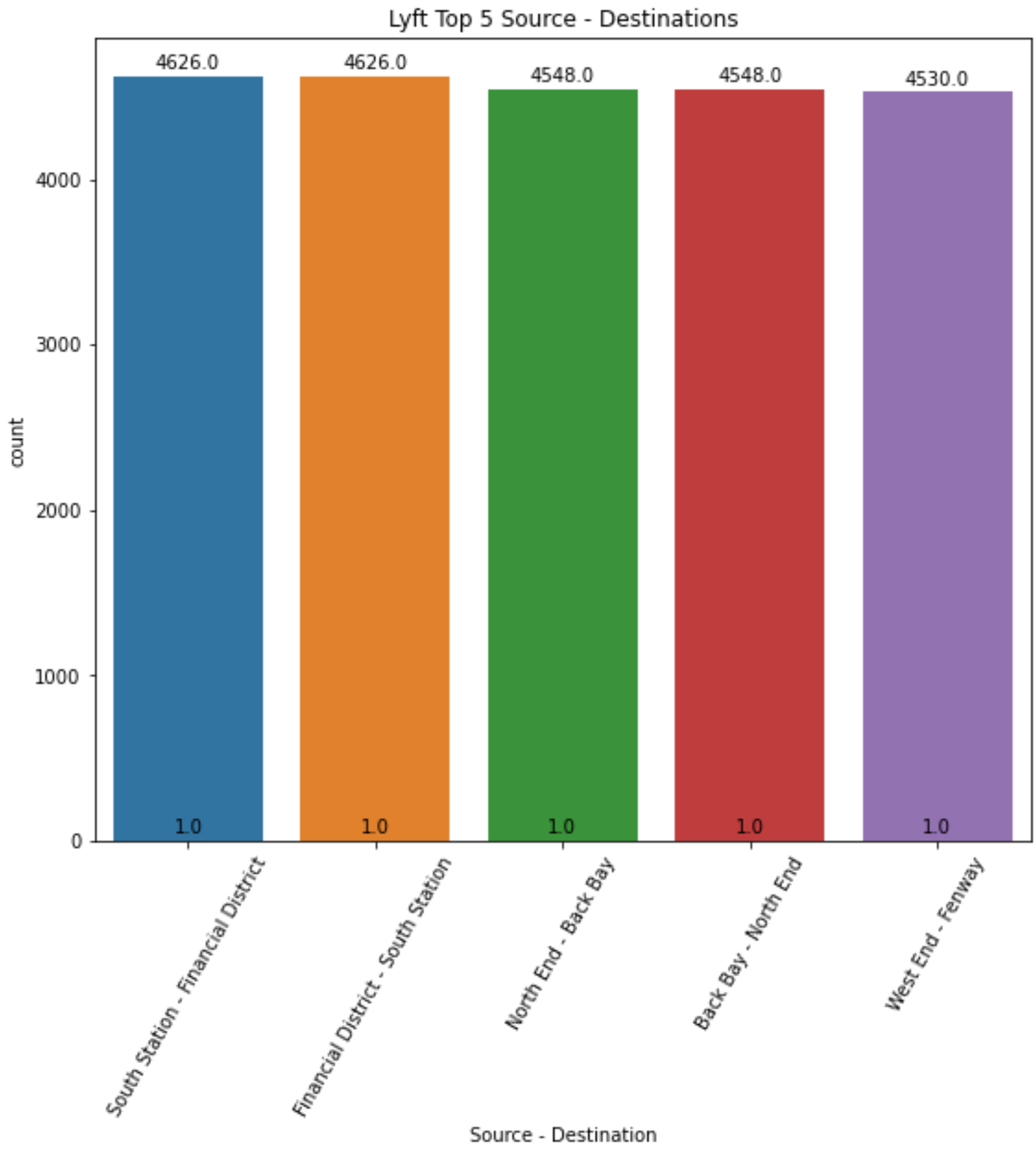
Mencari tempat awal dan tujuan 5 teratas dari jenis taksi Uber dan Lyft

In [22]:
```
uber_new_data = uber_data.groupby(['source','destination']).size().reset_
index()
uber_new_data.columns = ['source','destination','total']
uber_new_data.sort_values('total', ascending=False, inplace=True)
uber_top5 = uber_new_data.head(5)
uber_top5["Source - Destination"] = uber_new_data["source"] + " - " + ube
r_new_data["destination"]
uber_top5 = uber_top5[["Source - Destination", "total"]]

# Plotting bar plot
plt.figure(figsize=(20, 8))
plt.subplot(121)
bp =sns.barplot(data = uber_top5, x = "Source - Destination", y = "tota
l")
bp.set_title("Uber Top 5 Source - Destinations")
loc, labels = plt.xticks()
bp.set_xticklabels(labels, rotation=60);
ax = sns.countplot(x='Source - Destination', data=uber_top5)
for i in ax.patches:
    ax.annotate(format(i.get_height(),'0.1f'), (i.get_x() + i.get_width
()/2.,i.get_height()),
               ha='center', va='center', xytext=(0,7), textcoords='offset
points')
```


Uber Top 5 Source - Destinations
count
5000
4000
3000
2000
1000
0
4908.0
4908.0
4866.0
4866.0
4830.0
1.0
1.0
1.0
1.0
1.0
Financial District - South Station
South Station - Financial District
North End - Back Bay
Back Bay - North End
West End - Fenway
Source - Destination

In [23]:

```
lyft_new_data = lyft_data.groupby(['source','destination']).size().reset_
index()
lyft_new_data.columns = ['source','destination','total']
lyft_new_data.sort_values('total', ascending=False, inplace=True)
lyft_top5 = lyft_new_data.head(5)
lyft_top5["Source - Destination"] = lyft_new_data["source"] + " - " + lyf
t_new_data["destination"]
lyft_top5 = lyft_top5[["Source - Destination", "total"]]

# Plotting bar plot
plt.figure(figsize=(20, 8))
plt.subplot(121)
bp =sns.barplot(data = lyft_top5, x = "Source - Destination", y = "tota
l")
bp.set_title("Lyft Top 5 Source - Destinations")
loc, labels = plt.xticks()
bp.set_xticklabels(labels, rotation=60);
ax = sns.countplot(x='Source - Destination', data=lyft_top5)
for i in ax.patches:
    ax.annotate(format(i.get_height(),'0.1f'), (i.get_x() + i.get_width
()/2.,i.get_height()),
               ha='center', va='center', xytext=(0,7), textcoords='offset
points')
```



Eksplorasi Data Uber

In [24]:
```
uber = df[df['cab_type']=='Uber'].reset_index(drop=True)
uber = uber.drop(columns=['cab_type','source','destination'], axis=1)
uber.head(5)
```

Out[24]:

	index	hour	day	month	name	price	distance	surge_multiplier	short_summary	
0	12	22	30	11	UberXL	12.0	1.11	1.0	Overcast	cl
1	13	10	13	12	Black	16.0	1.11	1.0	Clear	c I
2	14	19	13	12	UberX	7.5	1.11	1.0	Mostly Cloudy	pa clo
3	15	23	16	12	WAV	7.5	1.11	1.0	Light Rain	
4	16	0	14	12	Black SUV	26.0	1.11	1.0	Overcast	cl

In [25]:
```
uber.describe()
```

Out[25]:

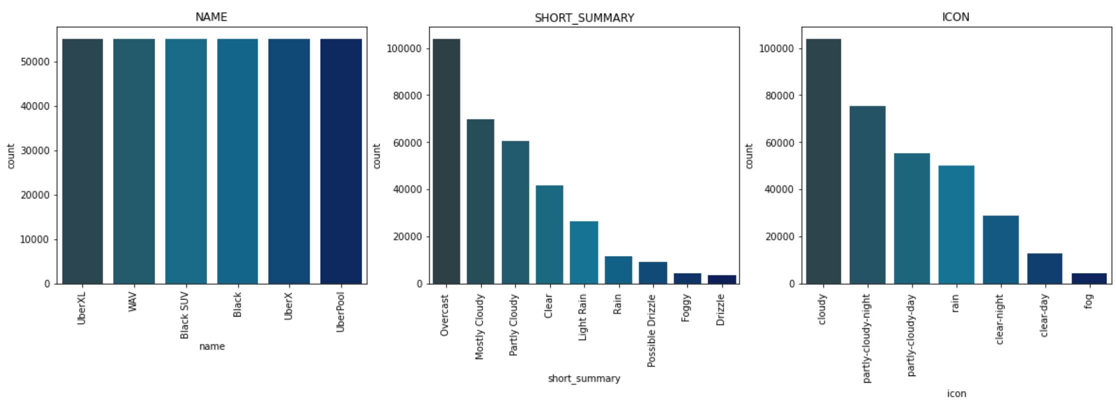
	index	hour	day	month	price	
count	330568.000000	330568.000000	330568.000000	330568.000000	330568.000000	33056
mean	347657.937060	11.608864	17.820176	11.586028	15.795343	2
std	200336.056904	6.942370	9.973335	0.492544	8.560300	1
min	12.000000	0.000000	1.000000	11.000000	4.500000	(
25%	174079.750000	6.000000	13.000000	11.000000	9.000000	1
50%	348082.500000	12.000000	17.000000	12.000000	12.500000	2
75%	520833.250000	18.000000	28.000000	12.000000	21.500000	2
max	693070.000000	23.000000	30.000000	12.000000	89.500000	7

In [26]:
```
cat_type_uber = uber.select_dtypes('object')
cat_type_uber.head()
```

Out[26]:

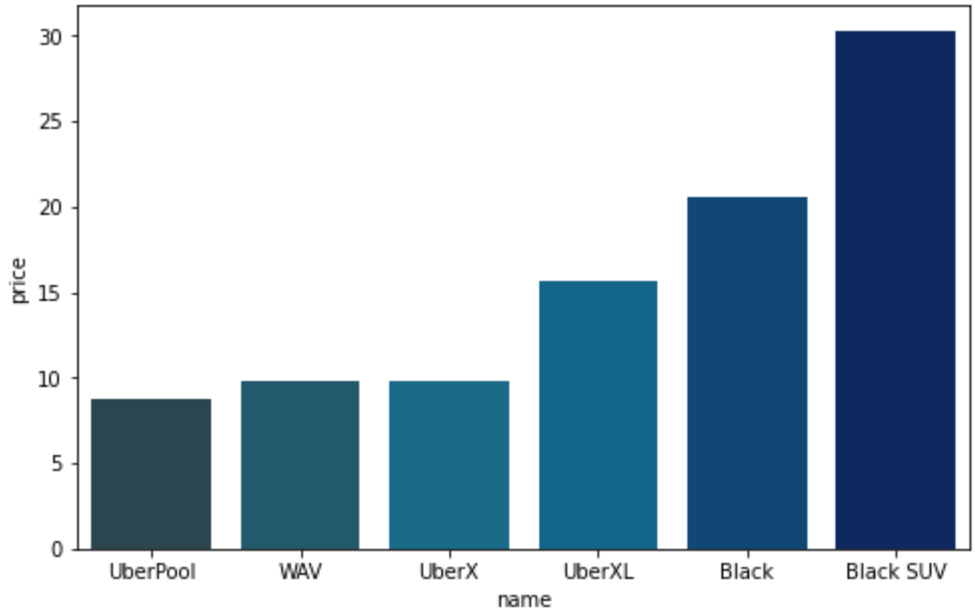
	name	short_summary	icon
0	UberXL	Overcast	cloudy
1	Black	Clear	clear-night
2	UberX	Mostly Cloudy	partly-cloudy-day
3	WAV	Light Rain	rain
4	Black SUV	Overcast	cloudy

In [27]:
```
row,col,i = 1,3,1
plt.figure(figsize = (20,5))
for cat_col in cat_type_uber.columns:
    plt.subplot(row,col,i)
    plt.title(cat_col.upper(), fontsize = 12)
    sns.countplot(cat_type_uber[cat_col], palette = "ocean_d", order= cat
_type_uber[cat_col].value_counts().index)
    plt.xticks(rotation = 90)
    i +=1
plt.show()
```



Transaksi berdasarkan **nama cab** pada Uber memiliki jumlah yang merata yaitu diatas nilai 50000, melalui visualisasi diagram diatas berdasarkan fitur **short_summary** jumlah transaksi tertinggi terjadi pada hari ketika mendung dengan jumlah diperkirakan lebih dari 100000 data dan transaksi terendah pada hari ketika mengalami gerimis dengan jumlah data kurang dari 20000.

In [28]:
```
level_price = uber.groupby(["name"])[["price"]].mean()
plt.figure(figsize = (8,5))
sns.barplot(level_price.index, level_price["price"],palette = "ocean_d",
            order = level_price["price"].sort_values().index)
plt.show()
```



Dari visualisasi di atas, dapat dilihat bahwa harga tertinggi pada cab jenis uber yaitu **Black SUV** dan level harga terendah adalah **UberPool**.

Eksplorasi Data Lyft

In [29]:
```python
lyft = df[df['cab_type']=='Lyft'].reset_index(drop=True)
lyft = lyft.drop(columns=['source','destination','cab_type'])
lyft.head()
```

Out[29]:

	index	hour	day	month	name	price	distance	surge_multiplier	short_summary	i
0	0	9	16	12	Shared	5.0	0.44	1.0	Mostly Cloudy	pa clo r
1	1	2	27	11	Lux	11.0	0.44	1.0	Rain	
2	2	1	28	11	Lyft	7.0	0.44	1.0	Clear	cl r
3	3	4	30	11	Lux Black XL	26.0	0.44	1.0	Clear	cl r
4	4	3	29	11	Lyft XL	9.0	0.44	1.0	Partly Cloudy	pa clo r

In [30]:
```python
lyft.describe()
```

Out[30]:

	index	hour	day	month	price	(
count	307408.000000	307408.000000	307408.000000	307408.000000	307408.000000	30740{
mean	345142.857069	11.628920	17.773477	11.587112	17.351396	2
std	199754.103330	6.955654	9.991441	0.492354	10.019171	1
min	0.000000	0.000000	1.000000	11.000000	2.500000	(
25%	172314.750000	6.000000	13.000000	11.000000	9.000000	1
50%	344388.500000	12.000000	17.000000	12.000000	16.500000	2
75%	518558.000000	18.000000	28.000000	12.000000	22.500000	2
max	693053.000000	23.000000	30.000000	12.000000	97.500000	6

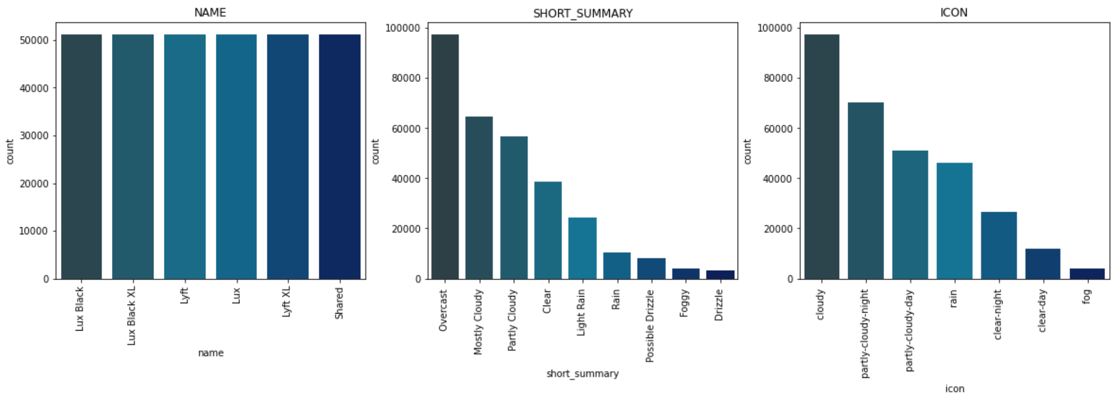
In [31]:
```python
cat_lyft = lyft.select_dtypes('object')
cat_lyft
```

Out[31]:

	name	short_summary	icon
0	Shared	Mostly Cloudy	partly-cloudy-night
1	Lux	Rain	rain
2	Lyft	Clear	clear-night
3	Lux Black XL	Clear	clear-night
4	Lyft XL	Partly Cloudy	partly-cloudy-night
...	...	...	...
307403	Lyft XL	Mostly Cloudy	partly-cloudy-night
307404	Lux	Mostly Cloudy	partly-cloudy-night
307405	Shared	Mostly Cloudy	partly-cloudy-night
307406	Lyft	Mostly Cloudy	partly-cloudy-night
307407	Lux Black XL	Mostly Cloudy	partly-cloudy-night

307408 rows × 3 columns

In [32]:
```python
row,col,i = 1,3,1
plt.figure(figsize = (20,5))
for cat_col in cat_lyft.columns:
    plt.subplot(row,col,i)
    plt.title(cat_col.upper(), fontsize = 12)
    sns.countplot(cat_lyft[cat_col], palette = "ocean_d", order= cat_lyft
[cat_col].value_counts().index)
    plt.xticks(rotation = 90)
    i +=1
plt.show()
```



In [33]:
```
level_price = lyft.groupby(["name"])[["price"]].mean()
plt.figure(figsize = (8,5))
sns.barplot(level_price.index, level_price["price"],palette = "ocean_d",
            order = level_price["price"].sort_values().index)
plt.show()
```

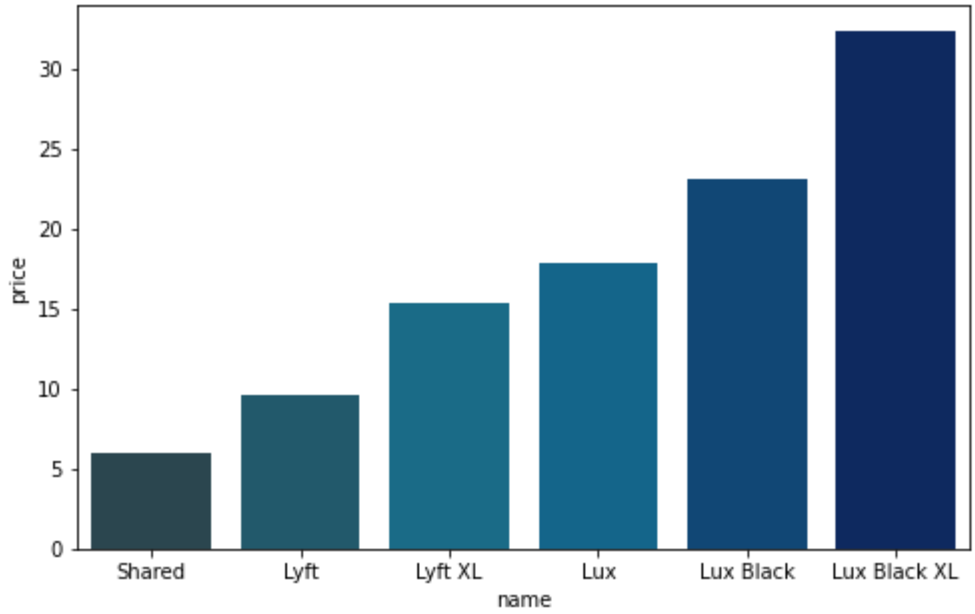


Diagram Bar diatas menunjukkan bahwa cab jenis **Lyft** dengan nama **Lux Black XL** memiliki level harga tertinggi dengan nilai diatas 30 sedangkan level harga terendah yaitu cab jenis **Lyft Shared**.

Data Preprocessing

In [34]:
```
# Binary encode kolom cab_type
df['cab_type'] = df['cab_type'].replace({'Lyft': 0, 'Uber': 1})
```

In [35]:
```
# Encoding semua kolom bertipe cateogory dengan onehot encoder
from sklearn.preprocessing import OneHotEncoder
categorical_cols = df.select_dtypes(include=['object','category']).column
s.tolist()

print(categorical_cols)
```

```
['source', 'destination', 'name', 'short_summary', 'icon']
```

In [36]:
```
# Inisiasi OneHotEncoder dan menggabungkan original dataframe dengan kolo
m encode ke dataframe
for col in categorical_cols:
        encoder = OneHotEncoder(handle_unknown='ignore')
        encoder_df = pd.DataFrame(encoder.fit_transform(df[[col]]).toarra
y())
        encoder_df.columns = encoder.get_feature_names_out([col])
        df = df.drop(col, axis=1)
        df = pd.concat([df, encoder_df], axis=1)
```

In [37]:
```
df.columns
```

Out[37]:
```
Index(['index', 'hour', 'day', 'month', 'cab_type', 'price', 'distance',
       'surge_multiplier', 'source_Back Bay', 'source_Beacon Hill',
       'source_Boston University', 'source_Fenway',
       'source_Financial District', 'source_Haymarket Square',
       'source_North End', 'source_North Station',
       'source_Northeastern University', 'source_South Station',
       'source_Theatre District', 'source_West End', 'destination_Back B
ay',
       'destination_Beacon Hill', 'destination_Boston University',
       'destination_Fenway', 'destination_Financial District',
       'destination_Haymarket Square', 'destination_North End',
       'destination_North Station', 'destination_Northeastern University
',
       'destination_South Station', 'destination_Theatre District',
       'destination_West End', 'name_Black', 'name_Black SUV', 'name_Lux
',
       'name_Lux Black', 'name_Lux Black XL', 'name_Lyft', 'name_Lyft XL
',
       'name_Shared', 'name_UberPool', 'name_UberX', 'name_UberXL', 'nam
e_WAV',
       'short_summary_ Clear ', 'short_summary_ Drizzle ',
       'short_summary_ Foggy ', 'short_summary_ Light Rain ',
       'short_summary_ Mostly Cloudy ', 'short_summary_ Overcast ',
       'short_summary_ Partly Cloudy ', 'short_summary_ Possible Drizzle
',
       'short_summary_ Rain ', 'icon_ clear-day ', 'icon_ clear-night ',
       'icon_ cloudy ', 'icon_ fog ', 'icon_ partly-cloudy-day ',
       'icon_ partly-cloudy-night ', 'icon_ rain '],
      dtype='object')
```

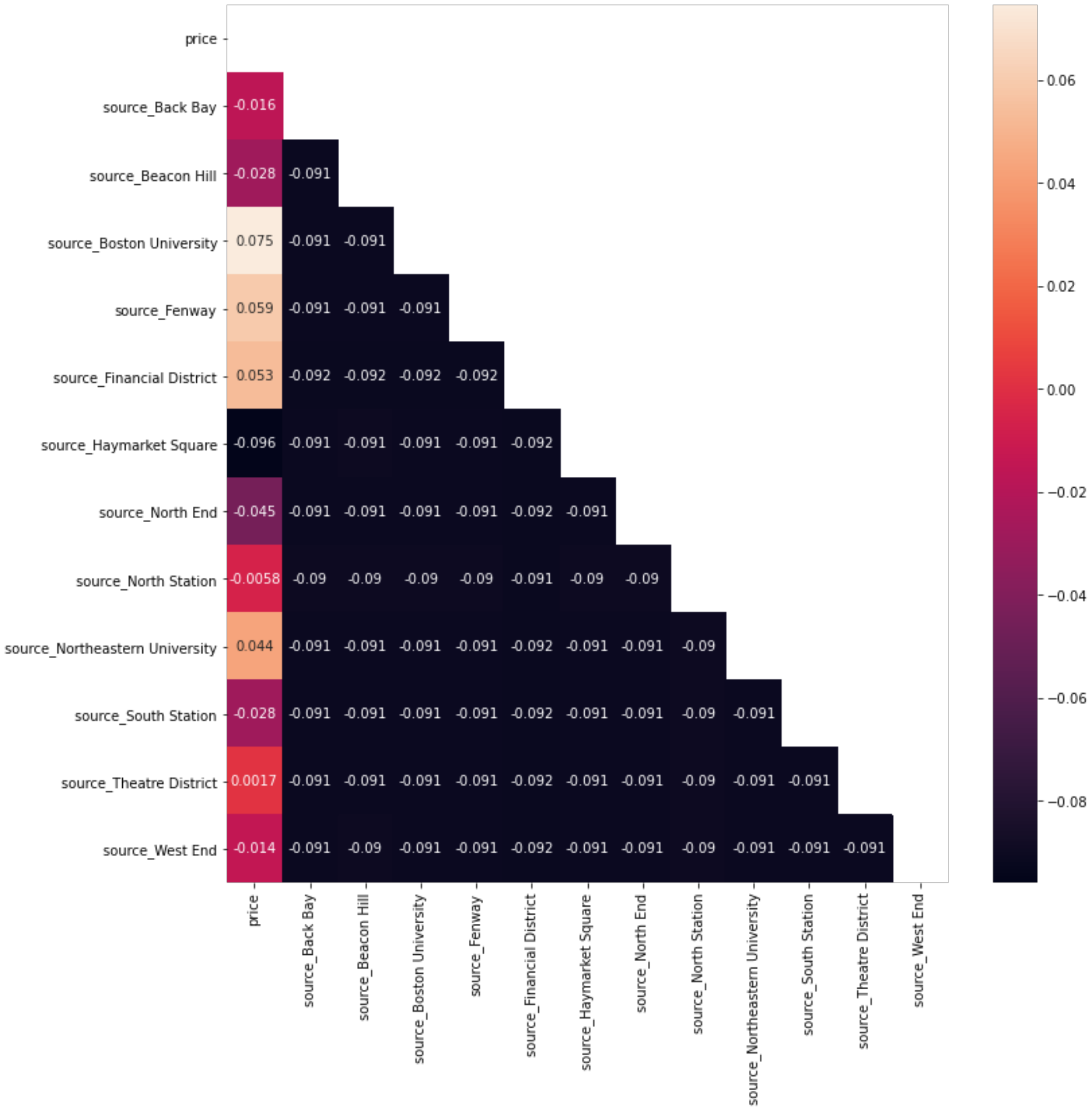
In [38]:
```
# Analisis dan cek korelasi antara price dengan kolom yang berhubungan de
ngan source
source_cols = ['price','source_Back Bay', 'source_Beacon Hill', 'source_B
oston University','source_Fenway',
               'source_Financial District', 'source_Haymarket Square','so
urce_North End', 'source_North Station',
               'source_Northeastern University', 'source_South Station','
source_Theatre District',
               'source_West End']
new_data = df[source_cols]

new_data.head()
```

Out[38]:

	price	source_Back Bay	source_Beacon Hill	source_Boston University	source_Fenway	source_Financial District
0	5.0	0.0	0.0	0.0	0.0	0.0
1	11.0	0.0	0.0	0.0	0.0	0.0
2	7.0	0.0	0.0	0.0	0.0	0.0
3	26.0	0.0	0.0	0.0	0.0	0.0
4	9.0	0.0	0.0	0.0	0.0	0.0

In [39]:

```
plt.figure(figsize=(12,12))
sns.heatmap(new_data.corr(), annot=True, mask=np.triu(new_data.corr()));
```



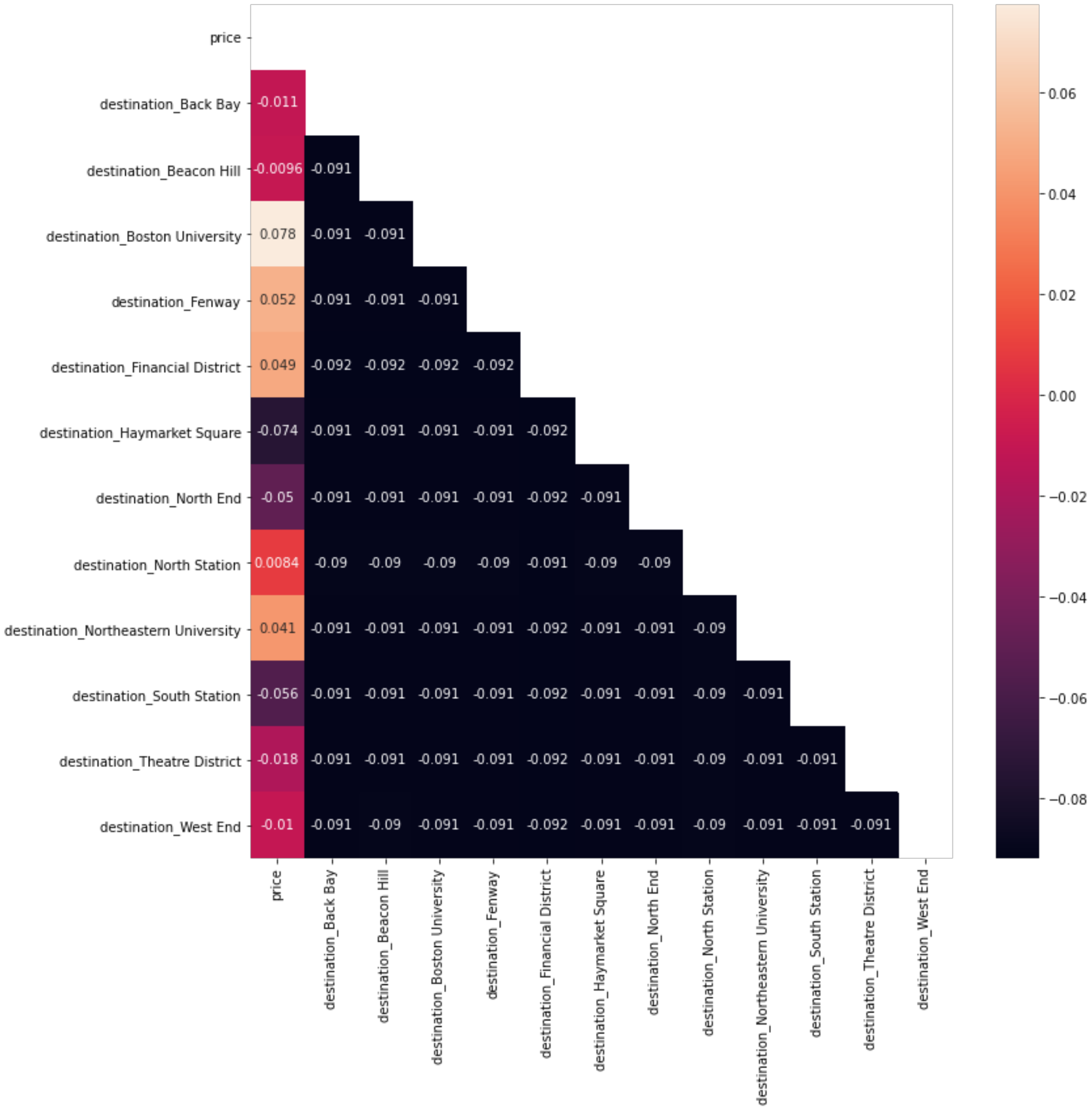
Kolom yang berhubungan dengan source memiliki pengaruh yang cukup signifikan terhadap price.

In [40]:
```
# Kemudian cek kolom destinasi dengan cara yang sama untuk melihat rate k
orelasi terhadap price
destination_cols =['price','destination_Back Bay','destination_Beacon Hil
l', 'destination_Boston University',
                   'destination_Fenway', 'destination_Financial District
','destination_Haymarket Square',
                   'destination_North End','destination_North Station', '
destination_Northeastern University',
                   'destination_South Station', 'destination_Theatre Dist
rict','destination_West End']
new_data = df[destination_cols]

new_data.head()
```

Out[40]:

	price	destination_Back Bay	destination_Beacon Hill	destination_Boston University	destination_Fenway	de
0	5.0	0.0	0.0	0.0	0.0	
1	11.0	0.0	0.0	0.0	0.0	
2	7.0	0.0	0.0	0.0	0.0	
3	26.0	0.0	0.0	0.0	0.0	
4	9.0	0.0	0.0	0.0	0.0	

In [41]:
```
plt.figure(figsize=(12,12))
sns.heatmap(new_data.corr(), annot=True, mask=np.triu(new_data.corr()));
```



Dari Plotting Heatmap diatas menunjukkan bahwa nilai korelasi dari source dan destination terhadap harga **sangat rendah**.

In [42]:
```
# Hapus kolom yang berkoreasi dengan source dan destination, kemudian mer
estrukturisasi dataframe
drop_cols = ['source_Back Bay', 'source_Beacon Hill', 'source_Boston Univ
ersity',
        'source_Fenway', 'source_Financial District', 'source_Haymarket Sq
uare',
        'source_North End', 'source_North Station',
        'source_Northeastern University', 'source_South Station',
        'source_Theatre District', 'source_West End', 'destination_Back Ba
y',
        'destination_Beacon Hill', 'destination_Boston University',
        'destination_Fenway', 'destination_Financial District',
        'destination_Haymarket Square', 'destination_North End',
        'destination_North Station', 'destination_Northeastern University
',
        'destination_South Station', 'destination_Theatre District',
        'destination_West End']
df = df.drop(drop_cols, axis=1)

print(df.shape)
df.head()
```

```
(637976, 36)
```

Out[42]:

	index	hour	day	month	cab_type	price	distance	surge_multiplier	name_Black	nam
0	0	9	16	12	0	5.0	0.44	1.0	0.0	
1	1	2	27	11	0	11.0	0.44	1.0	0.0	
2	2	1	28	11	0	7.0	0.44	1.0	0.0	
3	3	4	30	11	0	26.0	0.44	1.0	0.0	
4	4	3	29	11	0	9.0	0.44	1.0	0.0	

5 rows × 36 columns

In [43]:

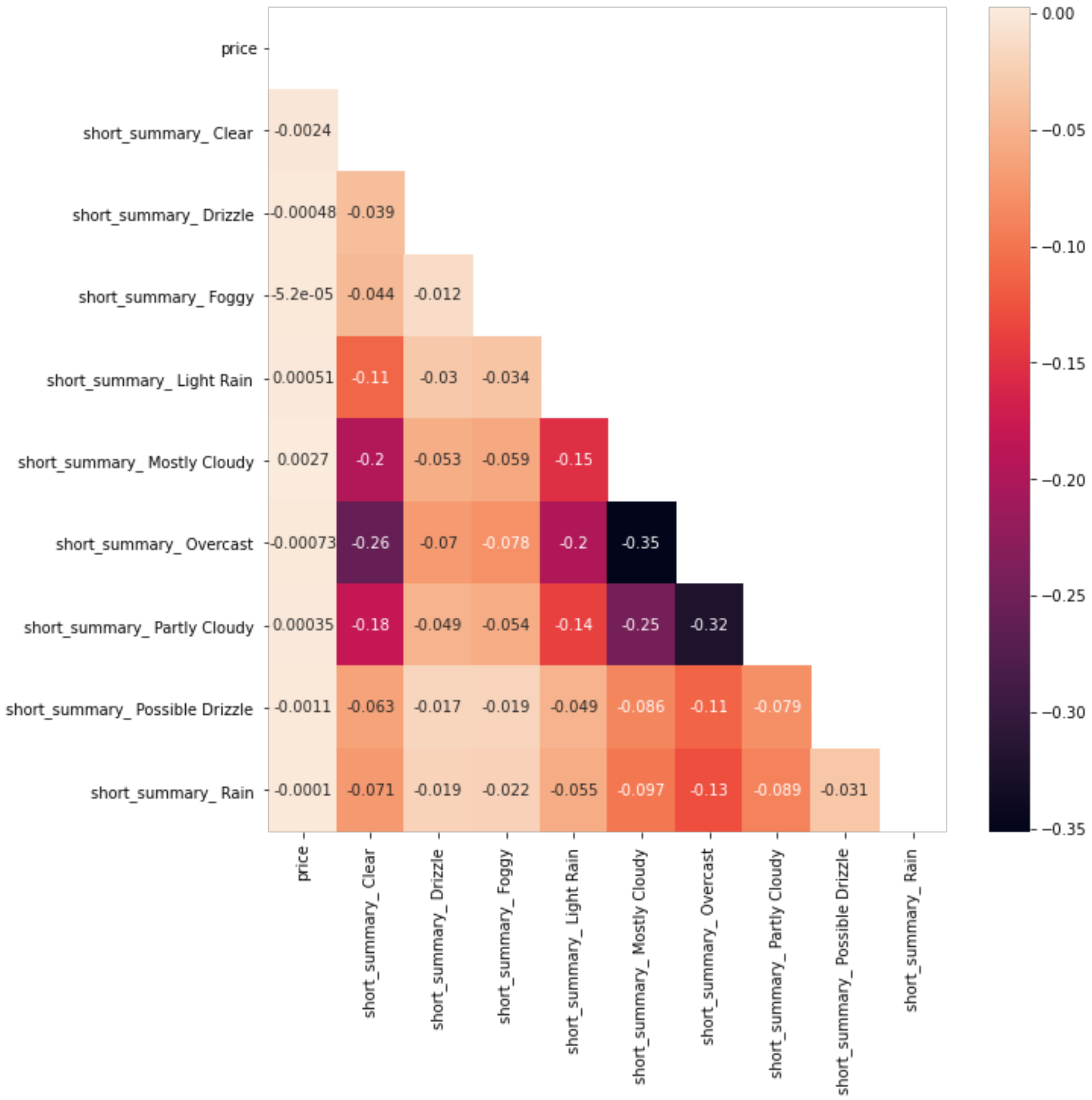
```python
# Cek korelasi dari kolom Summary dengan price
summary_cols = ['price','short_summary_ Clear ','short_summary_ Drizzle
', 'short_summary_ Foggy ',
                'short_summary_ Light Rain ', 'short_summary_ Mostly Clou
dy ','short_summary_ Overcast ',
                'short_summary_ Partly Cloudy ','short_summary_ Possible
Drizzle ', 'short_summary_ Rain ']
new_data = df[summary_cols]

new_data.head()
```

Out[43]:

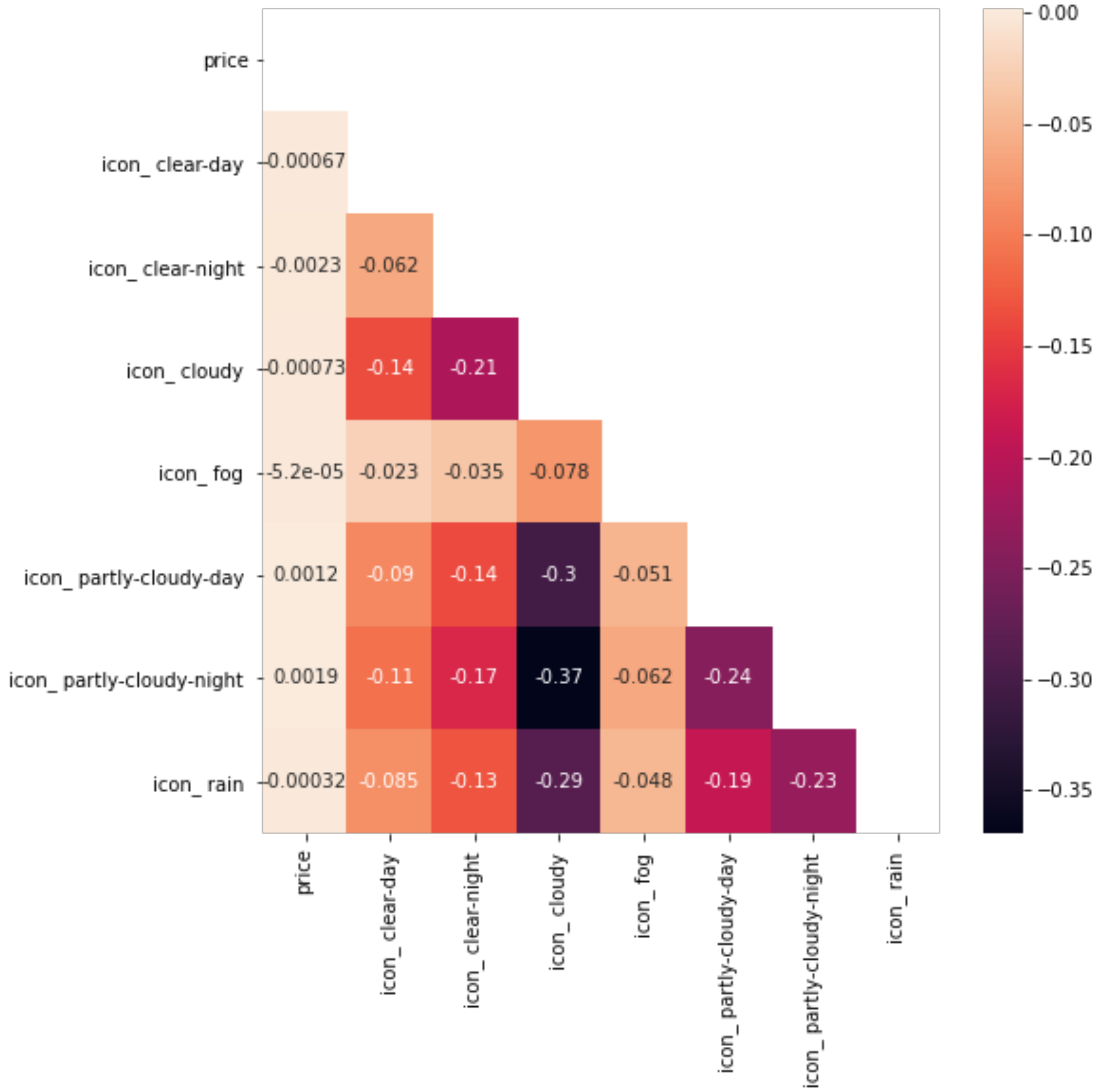
	price	short_summary_ Clear	short_summary_ Drizzle	short_summary_ Foggy	short_summary_ Light Rain	short_sumı Mostly C
0	5.0	0.0	0.0	0.0	0.0	
1	11.0	0.0	0.0	0.0	0.0	
2	7.0	1.0	0.0	0.0	0.0	
3	26.0	1.0	0.0	0.0	0.0	
4	9.0	0.0	0.0	0.0	0.0	

In [44]:

```python
plt.figure(figsize=(10,10))
sns.heatmap(new_data.corr(),annot=True,mask=np.triu(new_data.corr()));
```



```
In [45]: # Cek korelasi antara kolom Icon dan Price
         icon_cols= ['price','icon_ clear-day ', 'icon_ clear-night ', 'icon_ clou
         dy ', 'icon_ fog ',
                'icon_ partly-cloudy-day ', 'icon_ partly-cloudy-night ','icon_ ra
         in ']
         new_data = df[icon_cols]

         plt.figure(figsize=(8,8))
         sns.heatmap(new_data.corr(),annot=True,mask=np.triu(new_data.corr()));
```



Dapat dilihat bahwa kolom summary dan kolom Icon **tidak ada pengaruh** ke Price karena nilai korelasi mereka terlalu rendah (hampir 0).

In [46]:
```
# Drop kolom yang berkorelasi dengan kolom Icon
drop_cols = ['short_summary_ Clear ','short_summary_ Drizzle ', 'short_summary_ Foggy ',
            'short_summary_ Light Rain ', 'short_summary_ Mostly Cloudy ','short_summary_ Overcast ',
            'short_summary_ Partly Cloudy ','short_summary_ Possible Drizzle ', 'short_summary_ Rain ',
            'icon_ clear-day ', 'icon_ clear-night ', 'icon_ cloudy ', 'icon_ fog ','icon_ partly-cloudy-day ',
            'icon_ partly-cloudy-night ','icon_ rain ']
df = df.drop(drop_cols,axis=1)
print(df.shape)
df.head()
```

```
(637976, 20)
```

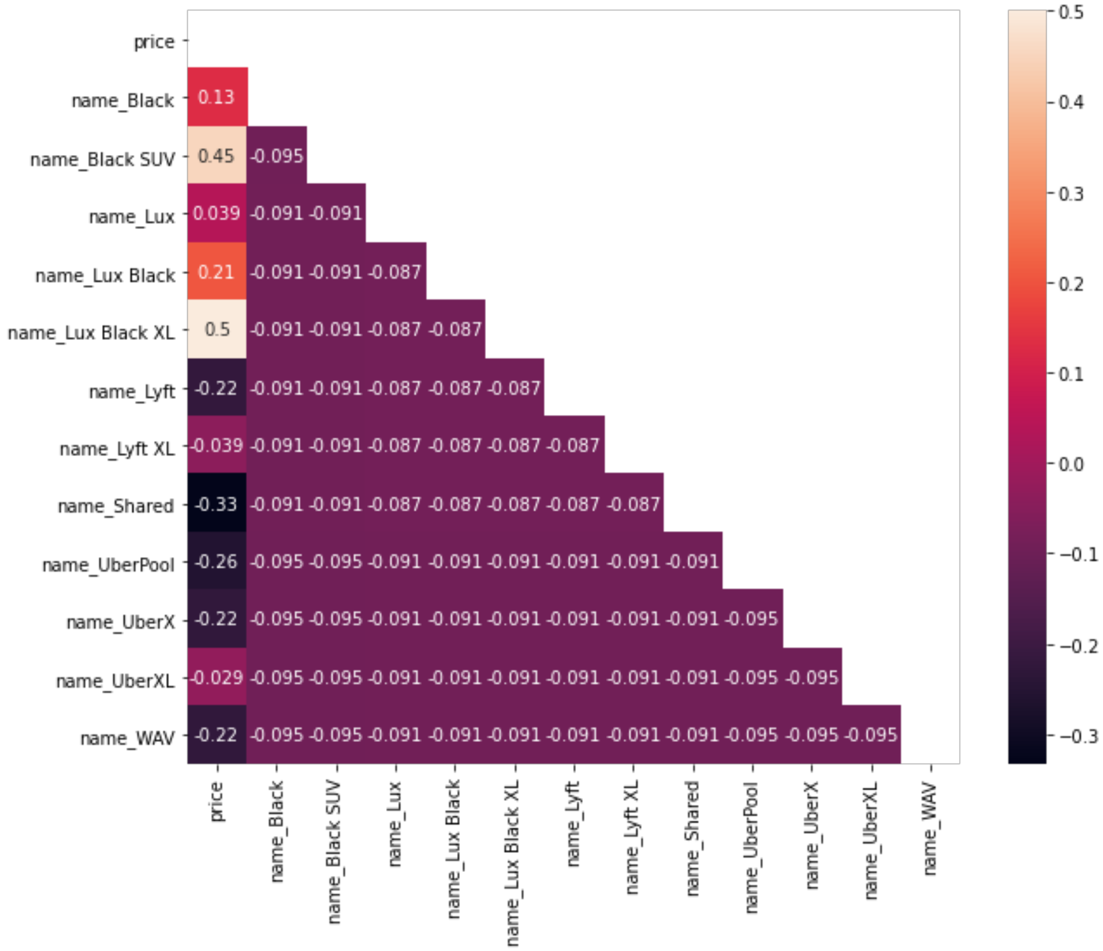
Out[46]:

	index	hour	day	month	cab_type	price	distance	surge_multiplier	name_Black	nam
0	0	9	16	12	0	5.0	0.44	1.0	0.0	
1	1	2	27	11	0	11.0	0.44	1.0	0.0	
2	2	1	28	11	0	7.0	0.44	1.0	0.0	
3	3	4	30	11	0	26.0	0.44	1.0	0.0	
4	4	3	29	11	0	9.0	0.44	1.0	0.0	

In [47]:
```
df.columns
```

Out[47]:
```
Index(['index', 'hour', 'day', 'month', 'cab_type', 'price', 'distance',
       'surge_multiplier', 'name_Black', 'name_Black SUV', 'name_Lux',
       'name_Lux Black', 'name_Lux Black XL', 'name_Lyft', 'name_Lyft XL
',
       'name_Shared', 'name_UberPool', 'name_UberX', 'name_UberXL',
       'name_WAV'],
      dtype='object')
```

In [48]:
```
# Analisis kolom Nama dengan Price
name_cols = ['price','name_Black', 'name_Black SUV','name_Lux', 'name_Lux Black', 'name_Lux Black XL', 'name_Lyft',
       'name_Lyft XL', 'name_Shared', 'name_UberPool','name_UberX', 'name_UberXL', 'name_WAV']
new_data = df[name_cols]

plt.figure(figsize=(10,8))
sns.heatmap(new_data.corr(),annot=True, mask=np.triu(new_data.corr()));
```

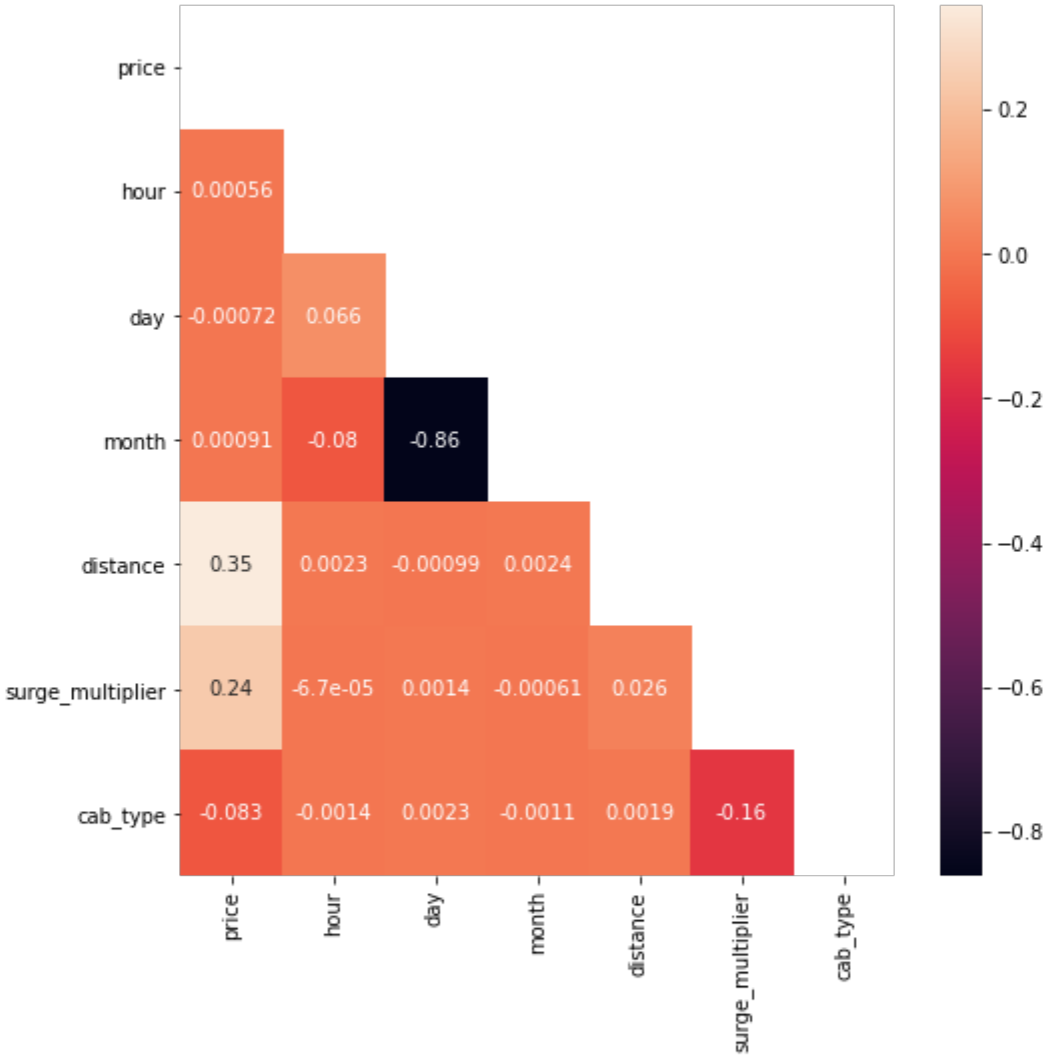


Beberapa nilai dari kolom Name **berpengaruh** terhadap nilai Price.

In [49]:
```
df.columns
```

Out[49]:
```
Index(['index', 'hour', 'day', 'month', 'cab_type', 'price', 'distance',
       'surge_multiplier', 'name_Black', 'name_Black SUV', 'name_Lux',
       'name_Lux Black', 'name_Lux Black XL', 'name_Lyft', 'name_Lyft XL
',
       'name_Shared', 'name_UberPool', 'name_UberX', 'name_UberXL',
       'name_WAV'],
      dtype='object')
```

In [50]:
```python
# Analisis sisa kolom
remaining_cols = ['price','hour', 'day', 'month', 'distance', 'surge_multiplier','cab_type']
new_data = df[remaining_cols]

plt.figure(figsize=(8,8))
sns.heatmap(new_data.corr(),annot=True,mask=np.triu(new_data.corr()));
```



Dari hasil analisis terhadap kolom-kolom tersebut bisa dilihat bahwa fitur hour, day, month memiliki korelasi yang **rendah**. Tetapi, kolom distance dan surge_multiplier memiliki korelasi yang **bagus** dengan price. Jadi drop kolom-kolom yang memiliki korelasi yang rendah.

In [51]:
```
# Drop kolom hour, day, dan month karena memiliki korelasi yang rendah te
rhadap price
df = df.drop(['hour', 'day', 'month'], axis=1)

print(df.shape)
df.head()
```

```
(637976, 17)
```

Out[51]:

	index	cab_type	price	distance	surge_multiplier	name_Black	name_Black SUV	name_Lux
0	0	0	5.0	0.44	1.0	0.0	0.0	0.0
1	1	0	11.0	0.44	1.0	0.0	0.0	1.0
2	2	0	7.0	0.44	1.0	0.0	0.0	0.0
3	3	0	26.0	0.44	1.0	0.0	0.0	0.0
4	4	0	9.0	0.44	1.0	0.0	0.0	0.0

In [52]:
```
# Cek nilai null pada semua fitur
df.isnull().sum()
```

Out[52]:
```
index                0
cab_type             0
price                0
distance             0
surge_multiplier     0
name_Black           0
name_Black SUV       0
name_Lux             0
name_Lux Black       0
name_Lux Black XL    0
name_Lyft            0
name_Lyft XL         0
name_Shared          0
name_UberPool        0
name_UberX           0
name_UberXL          0
name_WAV             0
dtype: int64
```

Cek outliers, cek nilai min dan max threshold. Kemudian plot kolom Price ke Box Plot

In [53]:
```
max_threshold = df['price'].quantile(0.99)
max_threshold
```

Out[53]:
```
42.5
```

In [54]:
```
df[df['price']>max_threshold]
```

Out[54]:

	index	cab_type	price	distance	surge_multiplier	name_Black	name_Black SUV	nan
645	706	0	52.5	3.25	2.00	0.0	0.0	
646	707	0	67.5	3.25	2.00	0.0	0.0	
706	769	0	45.5	4.76	1.00	0.0	0.0	
1005	1094	0	45.5	4.31	1.00	0.0	0.0	
1210	1318	0	45.5	5.33	1.00	0.0	0.0	
...	...	...	...	...	...	...	...	
637394	692439	1	47.0	5.56	1.00	0.0	1.0	
637637	692698	0	52.5	4.58	1.25	0.0	0.0	
637813	692891	0	47.5	5.42	1.00	0.0	0.0	
637878	692962	1	51.0	7.36	1.00	0.0	1.0	
637917	693007	1	49.5	7.36	1.00	0.0	1.0	

5589 rows × 17 columns

In [55]:
```
min_threshold = df['price'].quantile(0.01)
min_threshold
```

Out[55]: 3.5

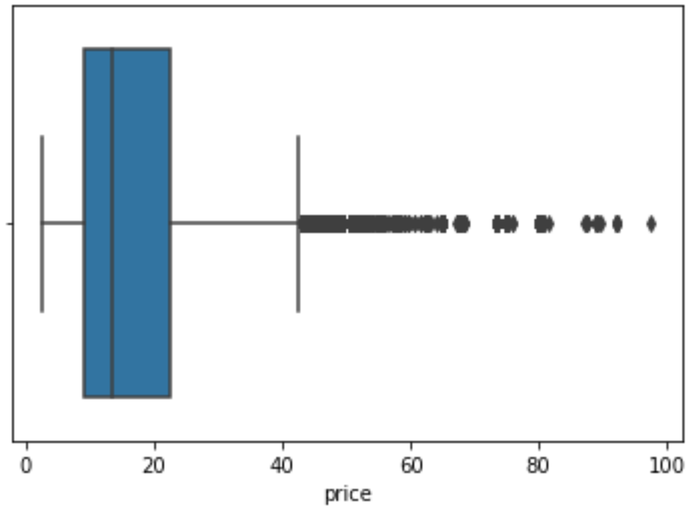
In [56]:
```
df[df['price']<min_threshold]
```

Out[56]:

	index	cab_type	price	distance	surge_multiplier	name_Black	name_Black SUV	nan
8	8	0	3.0	1.08	1.0	0.0	0.0	
50	53	0	3.0	0.71	1.0	0.0	0.0	
159	174	0	3.0	1.40	1.0	0.0	0.0	
312	336	0	3.0	1.02	1.0	0.0	0.0	
361	390	0	3.0	0.64	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	
637611	692670	0	3.0	1.69	1.0	0.0	0.0	
637660	692723	0	3.0	3.08	1.0	0.0	0.0	
637705	692772	0	3.0	0.70	1.0	0.0	0.0	
637779	692854	0	3.0	3.13	1.0	0.0	0.0	
637829	692908	0	3.0	1.42	1.0	0.0	0.0	

5754 rows × 17 columns

In [57]:
```
sns.boxplot(df['price']);
```



In [58]:
```
outliers = np.where(df['price']>42.5)

print(outliers[0])
print(np.count_nonzero(np.where(df['price']>42.5)))
```

```
[   645    646    706 ... 637813 637878 637917]
5589
```

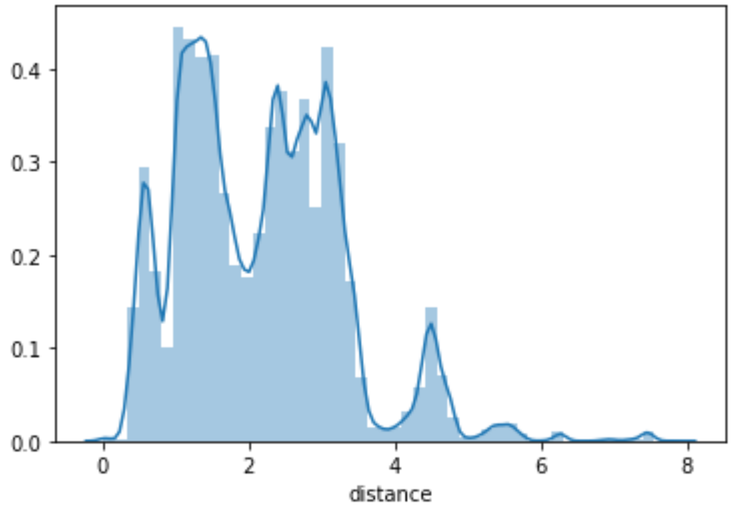
Hapus semua 5589 baris pada array tersebut, karena jika outliers tersebut disertakan, maka nilai error akan bertambah.

In [59]:
```
df.drop(outliers[0], inplace=True)
df.shape
```

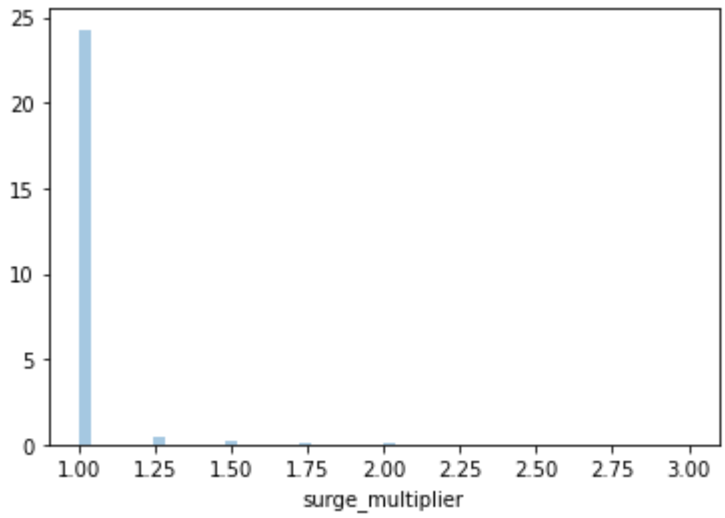
Out[59]:
```
(632387, 17)
```

In [60]:
```
# Cek Skewness pada semua fitur
from scipy.stats import skew
columns = ['distance', 'surge_multiplier']
for col in columns:
    print(col)
    print(skew(df[col]))
    plt.figure()
    sns.distplot(df[col]);
    plt.show()
```

```
distance
0.7767567965635372
```

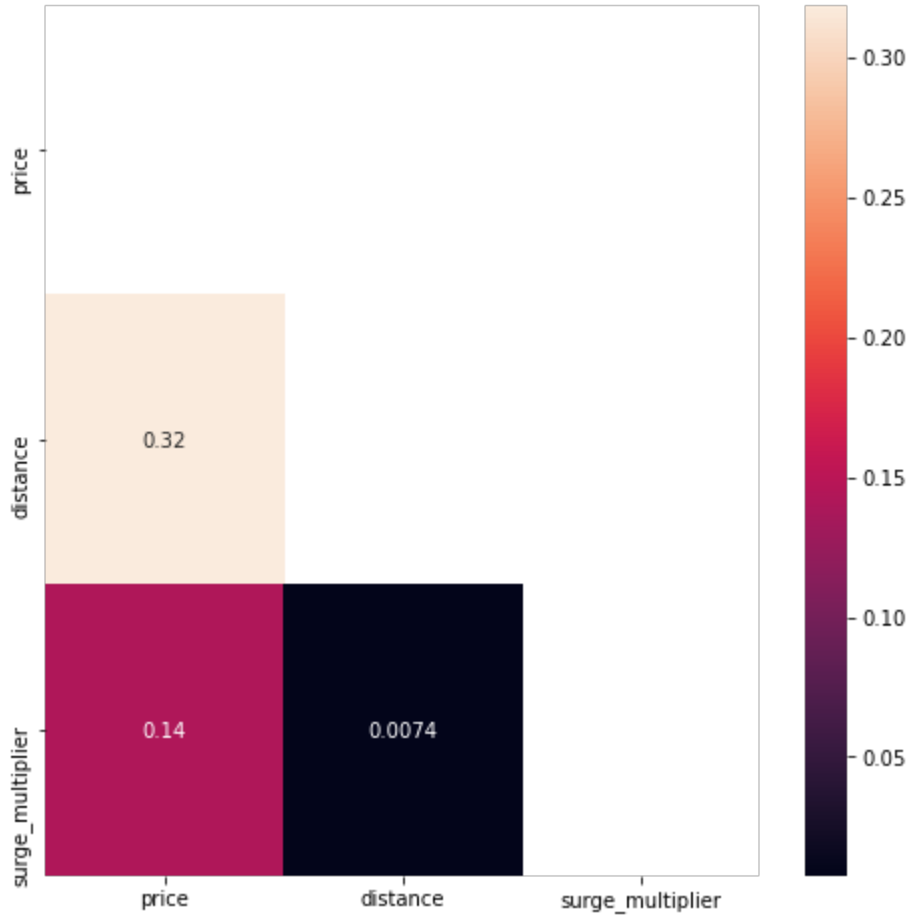



```
surge_multiplier
8.847590589161022
```



Kolom Distance dan Surge Multiplier memiliki skew yang sangat tinggi dengan nilai 0.77 dan 8.84.

In [61]:
```
#  Cek nilai korelasi pada kolom Distance dan Surge Multiplier terhadap p
redictand(price)
columns = ['price','distance','surge_multiplier']
new_df = df[columns]

plt.figure(figsize=(8,8))
sns.heatmap(new_df.corr(), annot=True, mask=np.triu(new_df.corr()));
```



Surge Multiplier memiliki nilai skew yang **tinggi** dan korelasi yang **kurang** dengan predictand (price), dimana kolom Distance memiliki korelasi yang **baik**. Jadi hapus skewneww dari Surge Multiplier menggunakan transformasi Box-Cox.Transformasi Box-Cox merupakan transformasi pangkat pada variabel respons yang dikembangkan oleh Box dan Cox, yang bertujuan untuk menormalkan data, melinearkan model regresi, dan menghomogenkan varians.

In [62]:
```
from scipy import stats
df['surge_multiplier'] = stats.boxcox(df['surge_multiplier'])[0]
pd.Series(df['surge_multiplier']).skew()
```

Out[62]: 5.64331840785854

Skewness dari kolom Surge_multiplier dikurangi dari 8.84 menjadi 5.64

In [63]:
```
df.price.describe()
```
Out[63]:
```
count    632387.000000
mean         16.245314
std           8.769536
min           2.500000
25%           9.000000
50%          13.500000
75%          22.500000
max          42.500000
Name: price, dtype: float64
```
In [64]:
```
df.columns
```
Out[64]:
```
Index(['index', 'cab_type', 'price', 'distance', 'surge_multiplier',
       'name_Black', 'name_Black SUV', 'name_Lux', 'name_Lux Black',
       'name_Lux Black XL', 'name_Lyft', 'name_Lyft XL', 'name_Shared',
       'name_UberPool', 'name_UberX', 'name_UberXL', 'name_WAV'],
      dtype='object')
```
In [65]:
```
df.drop(columns=['index'], axis=1, inplace=True)
```
In [66]:
```
df.columns
```
Out[66]:
```
Index(['cab_type', 'price', 'distance', 'surge_multiplier', 'name_Black
',
       'name_Black SUV', 'name_Lux', 'name_Lux Black', 'name_Lux Black X
L',
       'name_Lyft', 'name_Lyft XL', 'name_Shared', 'name_UberPool',
       'name_UberX', 'name_UberXL', 'name_WAV'],
      dtype='object')
```
In [67]:
```
# Rename Kolom
df.rename(columns={'name_Black':'Uber Black','name_Black SUV':'Uber Black
SUV', 'name_Lux':'Lyft Lux',
                   'name_Lux Black':'Lyft Lux Black','name_Lux Black XL
':'Lyft Lux Black XL','name_Lyft':'Lyft',
                   'name_Lyft XL':'Lyft XL', 'name_Shared':'Lyft Shared',
'name_UberPool':'Uber Pool',
                   'name_UberX':'Uber X', 'name_UberXL':'Uber XL', 'name_
WAV':'Uber WAV'}, inplace=True)
```
In [68]:
```
df.columns
```
Out[68]:
```
Index(['cab_type', 'price', 'distance', 'surge_multiplier', 'Uber Black
',
       'Uber Black SUV', 'Lyft Lux', 'Lyft Lux Black', 'Lyft Lux Black X
L',
       'Lyft', 'Lyft XL', 'Lyft Shared', 'Uber Pool', 'Uber X', 'Uber XL
',
       'Uber WAV'],
      dtype='object')
```

In [69]:
```
df.drop(columns=['cab_type'], inplace=True)
df.head()
```

Out[69]:

	price	distance	surge_multiplier	Uber Black	Uber Black SUV	Lyft Lux	Lyft Lux Black	Lyft Lux Black XL	Lyft	Lyft XL	Lyft Shared
0	5.0	0.44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1	11.0	0.44	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	7.0	0.44	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	26.0	0.44	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
4	9.0	0.44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

In [70]:
```
y = df['price']

y.head(3)
```

Out[70]:
```
0      5.0
1     11.0
2      7.0
Name: price, dtype: float64
```

In [71]:
```
X = df.drop(columns=['price'], axis=1)

X.head()
```

Out[71]:

	distance	surge_multiplier	Uber Black	Uber Black SUV	Lyft Lux	Lyft Lux Black	Lyft Lux Black XL	Lyft	Lyft XL	Lyft Shared	Uber Pool
0	0.44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	0.44	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.44	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
3	0.44	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

Modelling

Splitting Data Train dan Data Testing

In [72]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=5)
model_LR = LinearRegression()
model = model_LR.fit(x_train, y_train)
ypred = model.predict(x_test)
```

In [73]:
```
X.shape
```
Out[73]:
```
(632387, 14)
```

In [74]:
```
print(x_test.shape)
print(y_test.shape)
print(ypred.shape)
```
```
(63239, 14)
(63239,)
(63239,)
```

In [75]:
```
model.coef_
```
Out[75]:
```
array([ 2.67107329e+00,  6.37996232e+02, -2.55474366e+10, -2.55474366e+1
0,
       -2.55474366e+10, -2.55474366e+10, -2.55474366e+10, -2.55474366e+1
0,
       -2.55474366e+10, -2.55474366e+10, -2.55474366e+10, -2.55474366e+1
0,
       -2.55474366e+10, -2.55474366e+10])
```

In [76]:
```
model.intercept_
```
Out[76]:
```
25547436596.91638
```

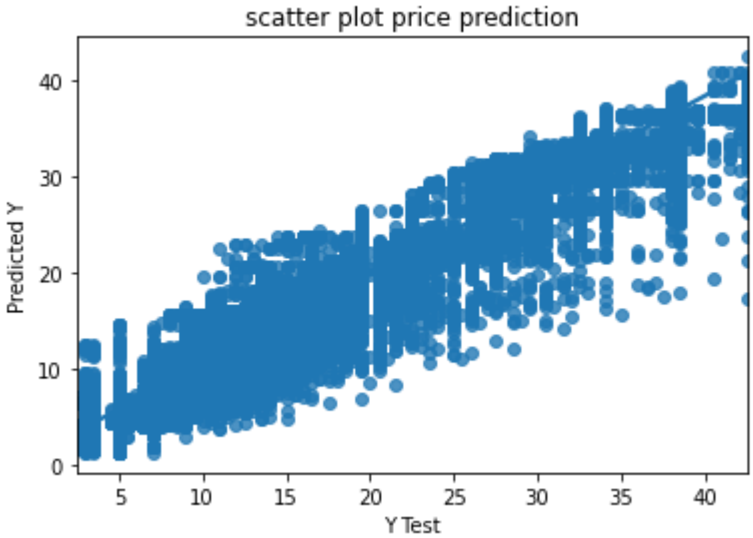
Evaluasi Model

In [77]:
```
# Cek nilai R2 untuk Linear Regression
from sklearn import metrics
metrics.r2_score(y_test, ypred)
```
Out[77]:
```
0.9322504646814097
```

Jadi tingkat akurasinya yaitu 0.93 atau 93%

In [78]:
```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cross_val = ShuffleSplit(n_splits=5, test_size=0.2, random_state=5)

cross_val_score(LinearRegression(), X, y, cv=cross_val)
```
Out[78]:
```
array([0.93109099, 0.93018927, 0.93080708, 0.93035974, 0.93018015])
```

In [79]:
```
sns.regplot(y_test, ypred)
plt.title("scatter plot price prediction");
plt.xlabel('Y Test');
plt.ylabel('Predicted Y');
```



In [80]:
```
# Menghitung akar dari mean squared error (MSE) untuk Linear Regression
mse = metrics.mean_squared_error(y_test,ypred)
rootmse = np.sqrt(mse)

print(mse)
print(rootmse)
```

```
5.207734475878149
2.2820461160717476
```

In [81]:
```
mae = metrics.mean_absolute_error(y_test, ypred)
rootmae = np.sqrt(mae)

print(mae)
print(rootmae)
```

```
1.682817449907354
1.2972345392824514
```

In [82]:
```
def predict_price(name_cab,distance,surge_multiplier):
    loc_index = np.where(X.columns==name_cab)[0][0]

    x = np.zeros(len(X.columns))
    x[0] = distance
    x[1] = surge_multiplier
    if loc_index >= 0:
        x[loc_index] = 1

    return model.predict([x])[0]
```

In [83]:
```
predict_price('Lyft Lux',0.44,0.0)
```

Out[83]:
```
12.545364379882812
```

In [84]:
```
predict_price('Lyft',0.44,0.0)
```
Out[84]: 4.429889678955078

In [85]:
```
predict_price('Uber WAV',0.44,0.0)
```
Out[85]: 5.086566925048828

In [86]:
```
predict_price('Lyft Shared',0.44, 0.0)
```
Out[86]: 1.3651084899902344

In [87]:
```
predict_price('Uber X',0.44, 0.0)
```
Out[87]: 5.085788726806641

In [88]:
```
predict_price('Lyft Lux Black',0.44, 0.0)
```
Out[88]: 17.669776916503906

In [89]:
```
predict_price('Lyft Lux Black XL',1.0, 0.0)
```
Out[89]: 27.791046142578125

In [90]:
```
predict_price('Uber Pool',1.0, 0.0)
```
Out[90]: 5.571388244628906

In [91]:
```
predict_price('Uber Black',1.0, 0.0)
```
Out[91]: 17.31753158569336

In [92]:
```
predict_price('Uber Black SUV',1.5, 0.0)
```
Out[92]: 28.22769546508789

In [93]:
```
# Simpan model kedalam file dengan pickle
import pickle
pickle.dump(model, open('./predict_price_model.pkl','wb'))
```

In [94]:
```
X.columns
```
Out[94]:
```
Index(['distance', 'surge_multiplier', 'Uber Black', 'Uber Black SUV',
       'Lyft Lux', 'Lyft Lux Black', 'Lyft Lux Black XL', 'Lyft', 'Lyft
XL',
       'Lyft Shared', 'Uber Pool', 'Uber X', 'Uber XL', 'Uber WAV'],
      dtype='object')
```

In [95]:
```
import json
columns = {
    'data_columns' : [col.lower() for col in X.columns]
}
with open("columns.json","w") as f:
    f.write(json.dumps(columns))
```

Kesimpulan

Dataset ini memiliki dimensi 693071 × 57, dengan begitu banyaknya fitur perlu diketahui fitur mana saja yang memiliki korelasi yang cukup, hal itu sangat berguna ketika dalam proses prediksi. Dikarenakan tujuan utama dalam projek ini adalah untuk **memprediksi harga** maka fitur price merupakan variabel dependent yang akan menjadi predictand. Dalam proses pemilihan fitur dapat dilakukan dengan menggunakan fungsi correlation dan juga bantuan visualisasi dari heatmap plot. Setelah dilakukan analisis dengan menggunakan fungsi korelasi dan heatmap plot dari 57 fitur kami mengambil fitur Distance, Surge_Multiplier dan Name_Cab karena fitur-fitur tersebut memiliki korelasi yang cukup berpengaruh ke variabel dependent(price).

Pada section EDA bisa dilihat bahwa Top 5 Source-Destination pada cab jenis Uber dan Lyft adalah sama, yaitu : Financial District-South Station (dan sebaliknya), Back Bay-North End (dan sebaliknya), West End-Fenway. Transaksi berdasarkan nama cab pada cab jenis Uber dan Lyft memiliki hasil yang sama tetapi beda jumlah nilai, berdasarkan fitur short_summary jumlah transaksi tertinggi terjadi pada hari ketika mendung data dan transaksi terendah pada hari ketika mengalami grimis. Harga tertinggi pada cab jenis Uber yaitu Black SUV dan level harga terendah adalah UberPool, sedangkan pada cab jenis Lyft dengan nama Lux Black XL memiliki level harga tertinggi dengan nilai diatas 30 sedangkan level harga terendah yaitu cab jenis Lyft Shared.

Untuk membuat model prediksi, pada projek ini menggunakan algoritma **Linear Regression**. Proses prediksi menggunakan R2 score dengan memanfaatkan library scikit-learn untuk mempermudah proses. R2 score merupakan salah satu metode yang digunakan untuk mengukur performa evaluasi pada regression. Hasil prediksi diatas dapat dilihat bahwa model prediksi menghasilkan nilai sebesar 0.93 atau 93% yang mana hasil tersebut menunjukkan nilai prediksi yang baik.