MSADS 31010 - Linear and Non-Linear Models Final Project (Team 3)

Used Cars Dataset in United States

Team members: Forough Mofidi, Naoki Tsumoto, Shaojie Chen, Bonny Mathew

In [1]:
```
#Import necessary packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Set Pandas options to display a maximum of 1000 rows
pd.set_option('display.max_rows', 1000)
```

EDA and Data-Preprocessing steps:

1. data distribution
2. check for missing values
3. impute missing values
4. conduct PCA for variables that are highly correlated

In [8]:
```
#from google.colab import drive
#drive.mount('/content/drive')
```

In [95]:
```
# Reading the dataset
loc = "/Users/foroughmofidi/Downloads/U of Chicago/Winter 2024/Linear Non-Linear/Final Project _ LNL/Data/used_cars.csv"

df = pd.read_csv(loc,
                 low_memory=False,
                 nrows= 100000,
                 verbose=True)
```

```
/var/folders/zx/0n2pn4p50s159drz1fg2x01w0000gn/T/ipykernel_91967/797224431.py:4: FutureWarning:

The 'verbose' keyword in pd.read_csv is deprecated and will be removed in a future version.
```

```
Tokenization took: 811.40 ms
Type conversion took: 1794.20 ms
Parser memory cleanup took: 7.03 ms
```

In [96]:
```
# Checking the snapshot of the data
df.head()
```

Out[96]:

	vin	back_legroom	bed	bed_height	bed_length	body_type	cabin	city	city_fuel_economy	combine_fuel_economy	daysonmarket	dealer_zip	description	engine_cylinders	engine_displacement
0	ZACNJABB5KPJ92081	35.1 in	NaN	NaN	NaN	SUV / Crossover	NaN	Bayamon	NaN	NaN	522	960	[!@@Additional Info@@!]Engine: 2.4L I4 ZERO EVAP M-AIR,Full Size Temporary Use Spare Tire,Manufacturer's Statement of Origin,Quick Order Package 2XB,Tires: 215/60R17 BSW AS Touring,Transmission: 9-Speed 948TE Automatic,Wheels: 17' x 7.0' Aluminum	I4	1300.0
1	SALCJ2FX1LH858117	38.1 in	NaN	NaN	NaN	SUV / Crossover	NaN	San Juan	NaN	NaN	207	922	[!@@Additional Info@@!]Keyless Entry,Ebony Morzine Headliner,Chrome Wheel Protection Pack,Powered Tailgate,Loadspace Mat,Wheels: 18' Style 5075 Gloss Sparkle Silver,Cargo Carrier,High Speed Emergency Braking,Adaptive Cruise Control w/Stop & Go,Sunshade,12-Way Electric Front Seats,Rubber Mats,Drive Pack,Basic Rear Seat Convenience Pack,Premium Interior Protection,Blind Spot Assist,Cargo Net	I4	2000.0
2	JF1VA2M67G9829723	35.4 in	NaN	NaN	NaN	Sedan	NaN	Guaynabo	17.0	NaN	1233	969	NaN	H4	2500.0
3	SALRR2RV0L2433391	37.6 in	NaN	NaN	NaN	SUV / Crossover	NaN	San Juan	NaN	NaN	196	922	[!@@Additional Info@@!]Fog Lights,7 Seat Package,Wheels: 21' 9 Spoke,GVWR: 6,900 lbs,Full Length Black Roof Rails,Twin-Speed Transfer Case,Cargo Carrier,Car Care Kit,Rubber Mat Set,Electronic Air Suspension,Prem Interior Protection/Storage Pack,Ebony Headlining,Wheel Protection Pack Chrome Locks,Cargo Mat,Tire Pressure Gauge,Windshield Sunshade,Basic Rear Seat Convenience Pack,Chrome Wheel Locks,Front Center Console Cooler Compartment,Tires: 21',Cabin Air Ionisation	V6	3000.0

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

	vin	back_legroom	bed	bed_height	bed_length	body_type	cabin	city	city_fuel_economy	combine_fuel_economy	daysonmarket	dealer_zip	description	engine_cylinders	engine_displacement
4	SALCJ2FXXLH862327	38.1 in	NaN	NaN	NaN	SUV / Crossover	NaN	San Juan	NaN	NaN	137	922	[!@@Additional Info@@!]Keyless Entry,Ebony Morzine Headliner,Chrome Wheel Protection Pack,ClearSight Rearview Mirror,Loadspace Mat,Wheels: 18' Style 5075 Gloss Sparkle Silver,Head-Up Display,Cargo Carrier,High Speed Emergency Braking,Adaptive Cruise Control w/Stop & Go,Sunshade,Technology Pack,12-Way Electric Front Seats,Rubber Mats,Drive Pack,Basic Rear Seat Convenience Pack,Fixed Panoramic Roof,Premium Interior Protection,Interactive Driver Display,Blind Spot Assist,Cargo Net,Wireless Devi...	I4	2000.0

In [97]:
```
# Checking the datatypes and dims of dataset
print('The dimensions of the dataset are:',df.shape)
df.info()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
The dimensions of the dataset are: (100000, 66)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 66 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   vin                      100000 non-null  object 
 1   back_legroom             95046 non-null   object 
 2   bed                      607 non-null     object 
 3   bed_height               8123 non-null    object 
 4   bed_length               8123 non-null    object 
 5   body_type                99599 non-null   object 
 6   cabin                    1667 non-null    object 
 7   city                     100000 non-null  object 
 8   city_fuel_economy        83984 non-null   float64
 9   combine_fuel_economy     0 non-null       float64
 10  daysonmarket             100000 non-null  int64  
 11  dealer_zip               100000 non-null  int64  
 12  description              97457 non-null   object 
 13  engine_cylinders         97002 non-null   object 
 14  engine_displacement      94661 non-null   float64
 15  engine_type              97002 non-null   object 
 16  exterior_color           98607 non-null   object 
 17  fleet                    56925 non-null   object 
 18  frame_damaged            56925 non-null   object 
 19  franchise_dealer         100000 non-null  bool   
 20  franchise_make           74584 non-null   object 
 21  front_legroom            95046 non-null   object 
 22  fuel_tank_volume         95046 non-null   object 
 23  fuel_type                97541 non-null   object 
 24  has_accidents            56925 non-null   object 
 25  height                   95046 non-null   object 
 26  highway_fuel_economy     83984 non-null   float64
 27  horsepower               94661 non-null   float64
 28  interior_color           88208 non-null   object 
 29  isCab                    56925 non-null   object 
 30  is_certified             0 non-null       float64
 31  is_cpo                   8730 non-null    object 
 32  is_new                   100000 non-null  bool   
 33  is_oemcpo                6110 non-null    object 
 34  latitude                 100000 non-null  float64
 35  length                   95046 non-null   object 
 36  listed_date              100000 non-null  object 
 37  listing_color            100000 non-null  object 
 38  listing_id               100000 non-null  int64  
 39  longitude                100000 non-null  float64
 40  main_picture_url         85008 non-null   object 
 41  major_options            93516 non-null   object 
 42  make_name                100000 non-null  object 
 43  maximum_seating          95046 non-null   object 
 44  mileage                  96443 non-null   float64
 45  model_name               100000 non-null  object 
 46  owner_count              54278 non-null   float64
 47  power                    85697 non-null   object 
 48  price                    100000 non-null  float64
 49  salvage                  56925 non-null   object 
 50  savings_amount           100000 non-null  int64  
 51  seller_rating            99358 non-null   float64
 52  sp_id                    100000 non-null  int64  
 53  sp_name                  100000 non-null  object 
 54  theft_title              56925 non-null   object 
 55  torque                   84650 non-null   object 
 56  transmission             98999 non-null   object 
                                    non-null   object 
```

```
 58  trimId                   96177 non-null   object
 59  trim_name                96159 non-null   object
 60  vehicle_damage_category  0 non-null       float64
 61  wheel_system             95376 non-null   object
 62  wheel_system_display     95376 non-null   object
 63  wheelbase                95046 non-null   object
 64  width                    95046 non-null   object
 65  year                     100000 non-null  int64
dtypes: bool(2), float64(13), int64(6), object(45)
memory usage: 49.0+ MB
```

In [98]:

```
# Check summary of numerical variables
df.describe()
```

Out[98]:

	city_fuel_economy	combine_fuel_economy	daysonmarket	dealer_zip	engine_displacement	highway_fuel_economy	horsepower	is_certified	latitude	listing_id	longitude	mileage	owner_count
count	83984.000000	0.0	100000.000000	100000.000000	94661.000000	83984.000000	94661.000000	0.0	100000.000000	1.000000e+05	100000.000000	96443.000000	54278.000000
mean	22.236700	NaN	77.682170	14346.712650	2802.220555	29.091696	245.387414	NaN	41.418281	2.752572e+08	-74.962234	31130.643437	1.454346
std	7.807983	NaN	109.629986	15502.216726	1159.202851	7.058758	86.521390	NaN	1.106694	8.943492e+06	3.935869	42835.631267	0.815929
min	8.000000	NaN	0.000000	922.000000	700.000000	11.000000	70.000000	NaN	18.346700	9.873062e+07	-122.320000	0.000000	1.000000
25%	18.000000	NaN	14.000000	6704.000000	2000.000000	25.000000	176.000000	NaN	40.755800	2.744280e+08	-74.331500	7.000000	1.000000
50%	21.000000	NaN	36.000000	7960.000000	2500.000000	28.000000	241.000000	NaN	41.126400	2.783952e+08	-73.830500	15236.000000	1.000000
75%	25.000000	NaN	82.000000	11743.000000	3500.000000	32.000000	295.000000	NaN	42.300500	2.803310e+08	-73.021800	43304.000000	2.000000
max	127.000000	NaN	2150.000000	98108.000000	8400.000000	127.000000	808.000000	NaN	47.549200	2.816772e+08	-66.078500	785778.000000	15.000000

In [99]:

```
# Check if the target variable (price) has any values below zero
df[df['price']< 0]
```

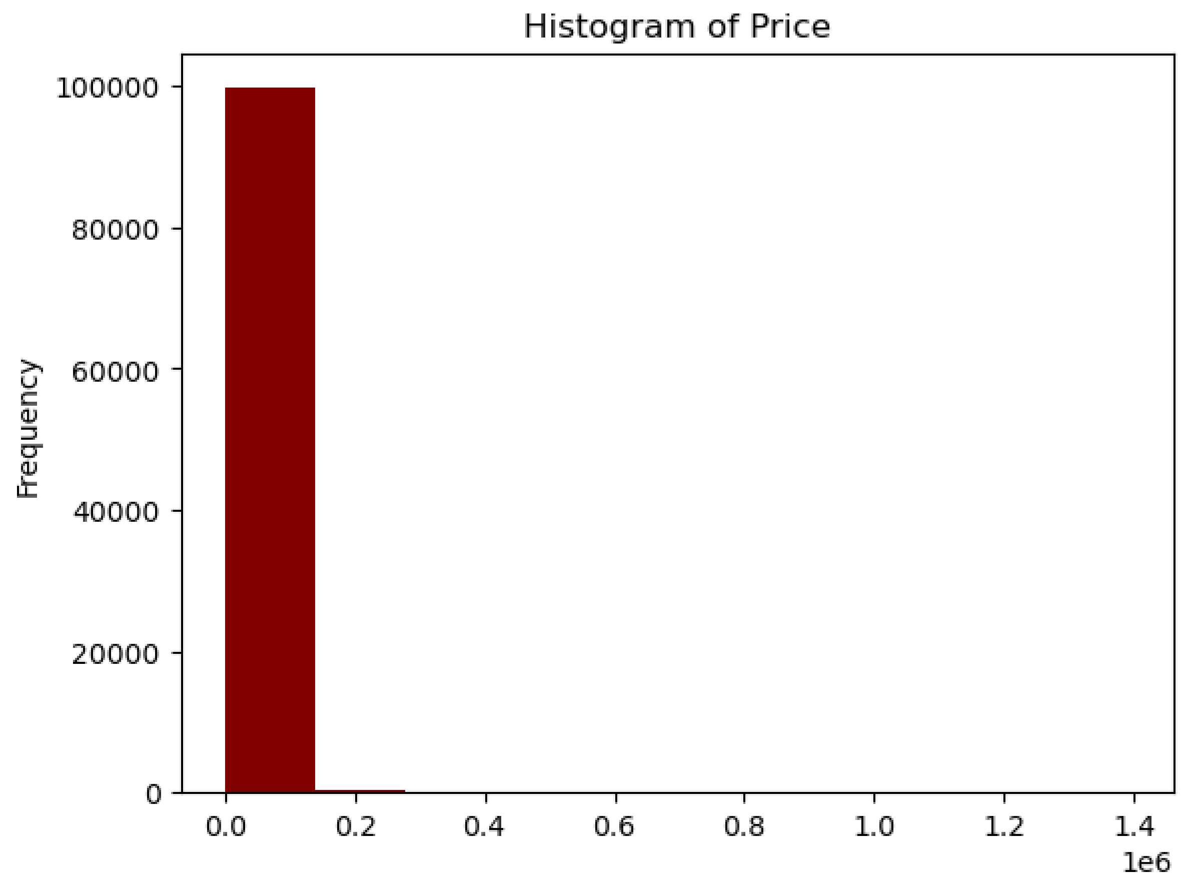
Out[99]:

vin	back_legroom	bed	bed_height	bed_length	body_type	cabin	city	city_fuel_economy	combine_fuel_economy	daysonmarket	dealer_zip	description	engine_cylinders	engine_displacement	engine_type	exterior_color	fleet

In [100…

```
# plot a histogram of the target variable: the plot shows a highly skewed distribution hence there is a need to scale
# the dataset by taking log

plt.hist(df['price'], color = 'maroon')
plt.title('Histogram of Price')
plt.ylabel('Frequency')
plt.show()
```

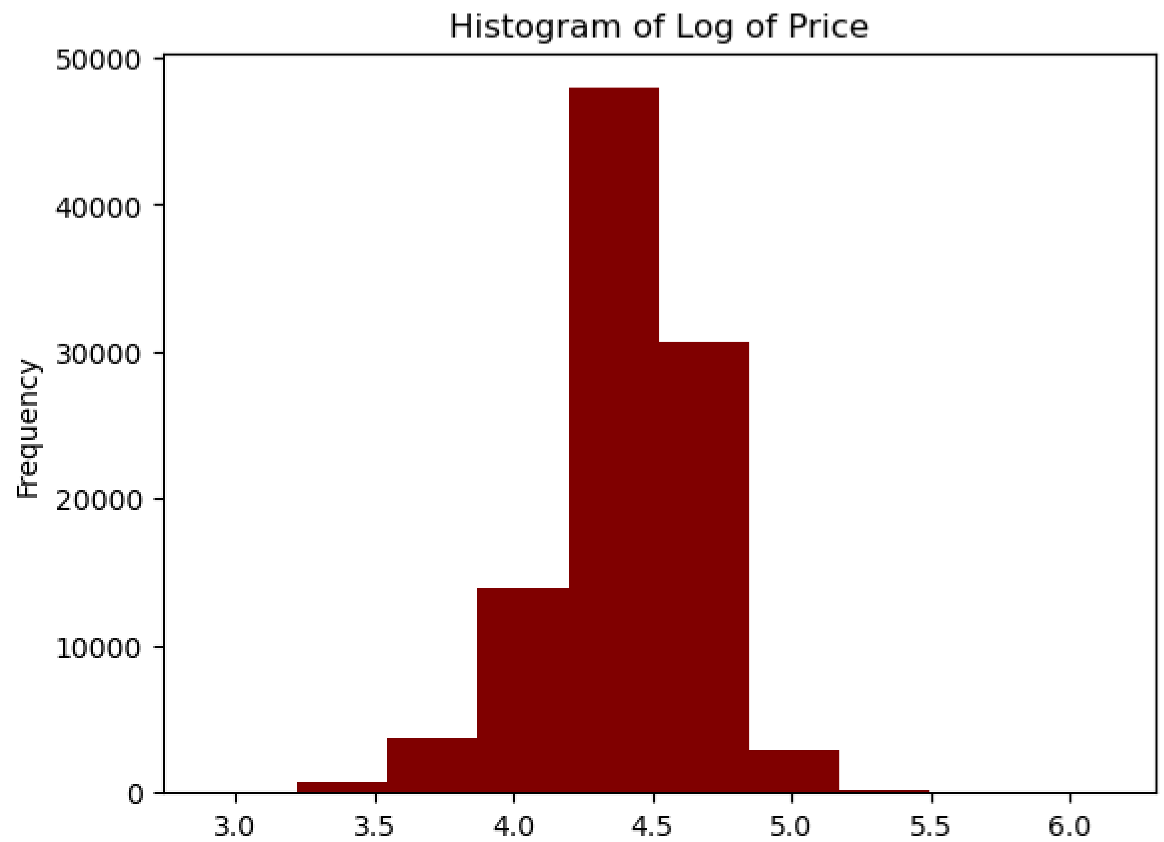
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [15]:
```
# Taking the log of the price variable
df['log10_price'] = df['price'].apply(lambda x: np.log10(x))
```

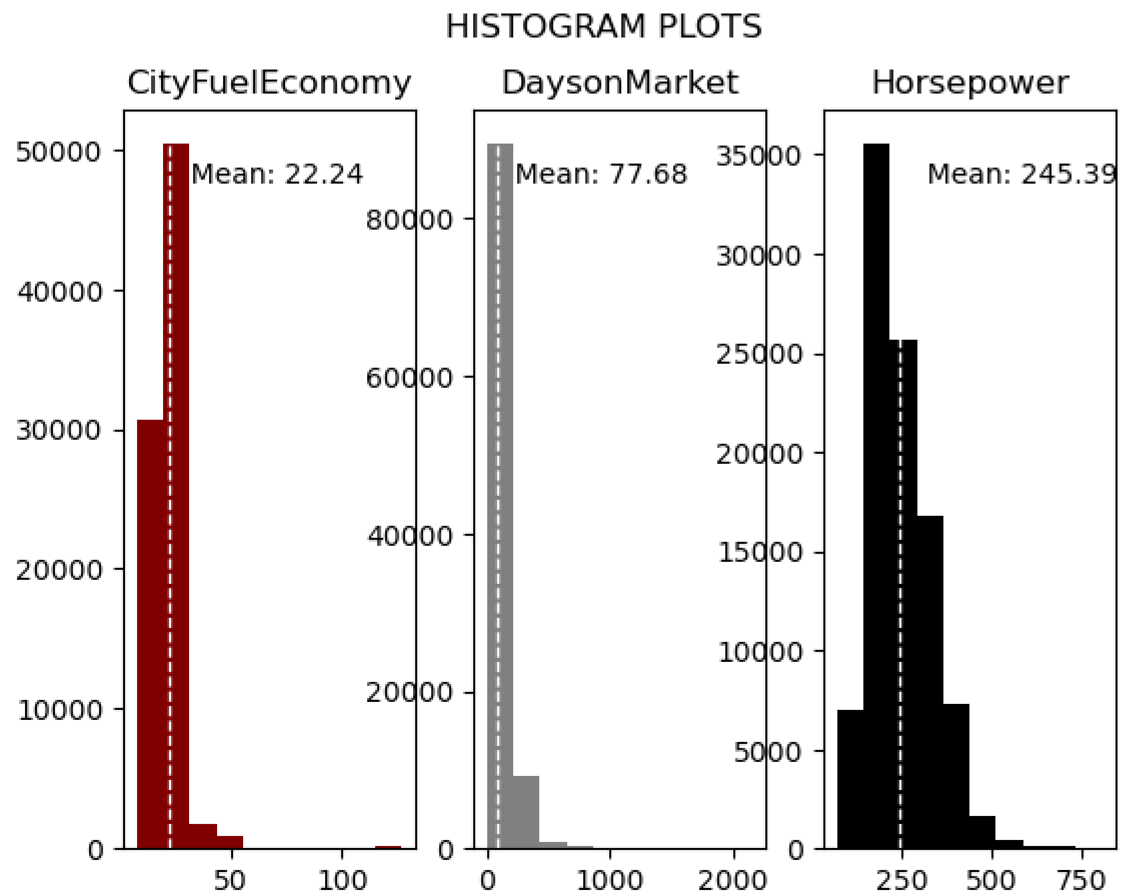
In [16]:
```
# plot a histogram of the transformed target variable

plt.hist(df['log10_price'], color = 'maroon')
plt.title('Histogram of Log of Price')
plt.ylabel('Frequency')
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [17]:
```
# histogram plots of few of the numerical variables

y = df['city_fuel_economy']
x = df['daysonmarket']
z = df['horsepower']
s = df['mileage']

plt.subplot(1, 3, 1)
plt.hist(y, color = 'maroon')
plt.axvline(y.mean(), color='w', linestyle='dashed', linewidth=1)
min_ylim, max_ylim = plt.ylim()
plt.text(y.mean()*1.1, max_ylim*0.9, '  Mean: {:.2f}'.format(y.mean()))
plt.title("CityFuelEconomy")

plt.subplot(1, 3, 2)
plt.hist(x, color = 'grey')
plt.axvline(x.mean(), color='w', linestyle='dashed', linewidth=1)
min_ylim, max_ylim = plt.ylim()
plt.text(x.mean()*1.1, max_ylim*0.9, '  Mean: {:.2f}'.format(x.mean()))
plt.title("DaysonMarket")

plt.subplot(1, 3, 3)
plt.hist(z, color = 'black')
plt.axvline(z.mean(), color='w', linestyle='dashed', linewidth=1)
min_ylim, max_ylim = plt.ylim()
plt.text(z.mean()*1.1, max_ylim*0.9, '  Mean: {:.2f}'.format(z.mean()))
plt.title("Horsepower")
plt.suptitle("HISTOGRAM PLOTS")
```

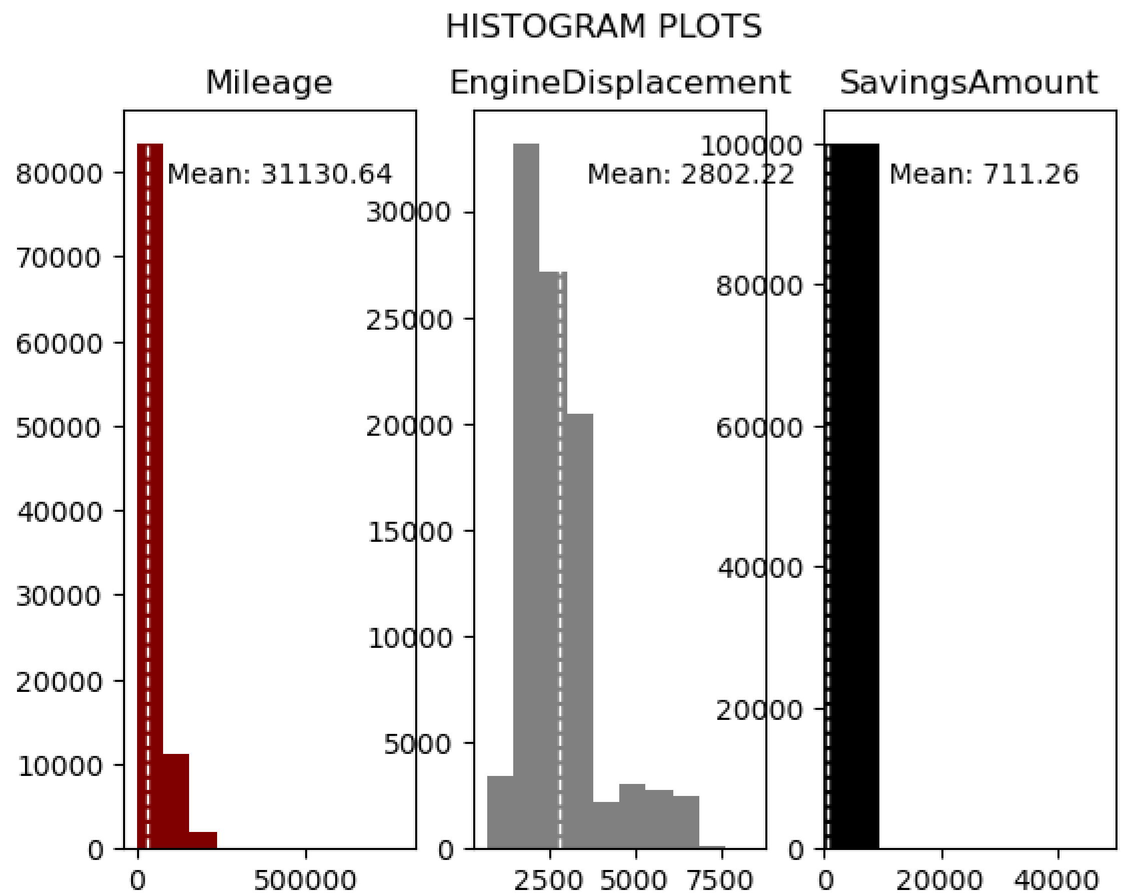
Out[17]:
```
Text(0.5, 0.98, 'HISTOGRAM PLOTS')
```



In [18]:
```
# histogram plots of few of the numerical variables

s = df['mileage']
t = df['engine_displacement']
u = df['savings_amount']


plt.subplot(1, 3, 1)
plt.hist(s, color = 'maroon')
plt.axvline(s.mean(), color='w', linestyle='dashed', linewidth=1)
min_ylim, max_ylim = plt.ylim()
plt.text(s.mean()*1.1, max_ylim*0.9, '  Mean: {:.2f}'.format(s.mean()))
plt.title("Mileage")

plt.subplot(1, 3, 2)
plt.hist(t, color = 'grey')
plt.axvline(t.mean(), color='w', linestyle='dashed', linewidth=1)
min_ylim, max_ylim = plt.ylim()
plt.text(t.mean()*1.1, max_ylim*0.9, '  Mean: {:.2f}'.format(t.mean()))
plt.title("EngineDisplacement")

plt.subplot(1, 3, 3)
plt.hist(u, color = 'black')
plt.axvline(u.mean(), color='w', linestyle='dashed', linewidth=1)
min_ylim, max_ylim = plt.ylim()
plt.text(u.mean()*1.1, max_ylim*0.9, '       Mean: {:.2f}'.format(u.mean()))
plt.title("SavingsAmount")
plt.xlim(0, 50000)
plt.suptitle("HISTOGRAM PLOTS")
```

Out[18]:
```
Text(0.5, 0.98, 'HISTOGRAM PLOTS')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Checking for Missing Values

In [19]:
```
# Identify if there are any variables with null values
df.isnull().sum().head()
```

Out[19]:
```
vin                0
back_legroom    4954
bed            99393
bed_height     91877
bed_length     91877
dtype: int64
```

In [20]:
```
# Identfify what percentage of the values are missing for each variable

null_percentage = df.isnull().sum()/df.shape[0]*100
null_percentage
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[20]:
```
vin                          0.000
back_legroom                 4.954
bed                         99.393
bed_height                  91.877
bed_length                  91.877
body_type                    0.401
cabin                       98.333
city                         0.000
city_fuel_economy           16.016
combine_fuel_economy       100.000
daysonmarket                 0.000
dealer_zip                   0.000
description                  2.543
engine_cylinders             2.998
engine_displacement          5.339
engine_type                  2.998
exterior_color               1.393
fleet                       43.075
frame_damaged               43.075
franchise_dealer             0.000
franchise_make              25.416
front_legroom                4.954
fuel_tank_volume             4.954
fuel_type                    2.459
has_accidents               43.075
height                       4.954
highway_fuel_economy        16.016
horsepower                   5.339
interior_color              11.792
isCab                       43.075
is_certified               100.000
is_cpo                      91.270
is_new                       0.000
is_oemcpo                   93.890
latitude                     0.000
length                       4.954
listed_date                  0.000
listing_color                0.000
listing_id                   0.000
longitude                    0.000
main_picture_url            14.992
major_options                6.484
make_name                    0.000
maximum_seating              4.954
mileage                      3.557
model_name                   0.000
owner_count                 45.722
power                       14.303
price                        0.000
salvage                     43.075
savings_amount               0.000
seller_rating                0.642
sp_id                        0.000
sp_name                      0.000
theft_title                 43.075
torque                      15.350
transmission                 1.001
transmission_display         1.001
trimId                       3.823
trim_name                    3.841
vehicle_damage_category    100.000
wheel_system                 4.624
wheel_system_display         4.624
```
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
width                    4.954
year                     0.000
log10_price              0.000
dtype: float64
```

In [21]:
```
# Identifying variables to drop
cols_todrop = null_percentage[null_percentage > 50].keys()
cols_todrop
```

Out[21]:
```
Index(['bed', 'bed_height', 'bed_length', 'cabin', 'combine_fuel_economy',
       'is_certified', 'is_cpo', 'is_oemcpo', 'vehicle_damage_category'],
      dtype='object')
```

In [22]:
```
# Dropping variables with more than 50% missing values
df_clean = df.drop(['price'], axis = 1)
df_clean = df_clean.drop(cols_todrop, axis=1)
df_clean.head()
```

Out[22]:

	vin	back_legroom	body_type	city	city_fuel_economy	daysonmarket	dealer_zip	description	engine_cylinders	engine_displacement	...	transmission	transmission_display	trimId	trim_name	wheel_sy
0	ZACNJABB5KPJ92081	35.1 in	SUV / Crossover	Bayamon	NaN	522	960	[!@@Additional Info@@!]Engine: 2.4L I4 ZERO EV...	I4	1300.0	...	A	9-Speed Automatic Overdrive	t83804	Latitude FWD	
1	SALCJ2FX1LH858117	38.1 in	SUV / Crossover	San Juan	NaN	207	922	[!@@Additional Info@@!]Keyless Entry,Ebony Mor...	I4	2000.0	...	A	9-Speed Automatic Overdrive	t86759	S AWD	
2	JF1VA2M67G9829723	35.4 in	Sedan	Guaynabo	17.0	1233	969	NaN	H4	2500.0	...	M	6-Speed Manual	t58994	Base	
3	SALRR2RV0L2433391	37.6 in	SUV / Crossover	San Juan	NaN	196	922	[!@@Additional Info@@!]Fog Lights,7 Seat Packa...	V6	3000.0	...	A	8-Speed Automatic Overdrive	t86074	V6 HSE AWD	
4	SALCJ2FXXLH862327	38.1 in	SUV / Crossover	San Juan	NaN	137	922	[!@@Additional Info@@!]Keyless Entry,Ebony Mor...	I4	2000.0	...	A	9-Speed Automatic Overdrive	t86759	S AWD	

5 rows × 57 columns

Dealing with missing character variables

In [23]:
```
df_obj = df_clean.select_dtypes(include=['object'])
#dropping character variables that are either ids or ones with too much granularity
df_obj = df_obj.drop(['description','vin','main_picture_url','trimId','major_options','torque'], axis = 1)
cols = df_obj.columns

#convert variables to categorical
df_obj[cols] = df[cols].astype('category')
```

In [24]:
```
df_obj.isnull().sum().head()
```

Out[24]:
```
back_legroom        4954
body_type            401
city                   0
engine_cylinders    2998
engine_type         2998
dtype: int64
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [25]:
```
!pip install feature-engine
```

```
Collecting feature-engine
  Obtaining dependency information for feature-engine from https://files.pythonhosted.org/packages/62/60/77fcc9d3cfaabab34027aa8ea0025c5e2d4cf9561fa9725a38f0785b43aa/feature_engine-1.6.2-py2.p
y3-none-any.whl.metadata
  Downloading feature_engine-1.6.2-py2.py3-none-any.whl.metadata (8.8 kB)
Requirement already satisfied: numpy>=1.18.2 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from feature-engine) (1.24.3)
Requirement already satisfied: pandas>=1.0.3 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from feature-engine) (2.2.0)
Requirement already satisfied: scikit-learn>=1.0.0 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from feature-engine) (1.3.0)
Requirement already satisfied: scipy>=1.4.1 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from feature-engine) (1.11.1)
Requirement already satisfied: statsmodels>=0.11.1 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from feature-engine) (0.14.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from pandas>=1.0.3->feature-engine) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from pandas>=1.0.3->feature-engine) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from pandas>=1.0.3->feature-engine) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from scikit-learn>=1.0.0->feature-engine) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from scikit-learn>=1.0.0->feature-engine) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from statsmodels>=0.11.1->feature-engine) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from statsmodels>=0.11.1->feature-engine) (23.1)
Requirement already satisfied: six in /Users/foroughmofidi/anaconda3/lib/python3.11/site-packages (from patsy>=0.5.2->statsmodels>=0.11.1->feature-engine) (1.16.0)
Downloading feature_engine-1.6.2-py2.py3-none-any.whl (328 kB)
   ---------------------------------------- 328.9/328.9 kB 615.7 kB/s eta 0:00:00a 0:00:01
Installing collected packages: feature-engine
Successfully installed feature-engine-1.6.2
```

In [26]:
```python
from feature_engine.imputation import CategoricalImputer
```

In [27]:
```python
imputer = CategoricalImputer()
```

In [28]:
```python
df_catg = imputer.fit_transform(df_obj)
```

In [29]:
```python
df_catg.isnull().sum().head()
```

Out[29]:
```
back_legroom       0
body_type          0
city               0
engine_cylinders   0
engine_type        0
dtype: int64
```

Dealing with missing numerical variables

In [30]:
```python
df_cleanNum = df_clean.select_dtypes(include=np.number)
df_cleanNum
```

Out[30]:

	city_fuel_economy	daysonmarket	dealer_zip	engine_displacement	highway_fuel_economy	horsepower	latitude	listing_id	longitude	mileage	owner_count	savings_amount	seller_rating	sp_id	year	log10_price
0	NaN	522	960	1300.0	NaN	177.0	18.3988	237132766	-66.1582	7.0	NaN	0	2.800000	370599	2019	4.364382
1	NaN	207	922	2000.0	NaN	246.0	18.4439	265946296	-66.0785	8.0	NaN	0	3.000000	389227	2020	4.667453
2	17.0	1233	969	2500.0	23.0	305.0	18.3467	173473508	-66.1098	NaN	3.0	0	NaN	370467	2016	4.672052
3	NaN	196	922	3000.0	NaN	340.0	18.4439	266911050	-66.0785	11.0	NaN	0	3.000000	389227	2020	4.828853
4	NaN	137	922	2000.0	NaN	246.0	18.4439	270957414	-66.0785	7.0	NaN	0	3.000000	389227	2020	4.689131
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99995	16.0	55	49091	6200.0	22.0	420.0	41.7981	276733070	-85.4287	NaN	NaN	0	5.000000	49994	2020	4.703962
99996	19.0	187	2920	2000.0	25.0	240.0	41.7982	267873284	-71.4481	67989.0	1.0	1764	4.000000	273779	2016	4.184691
99997	24.0	34	1776	2000.0	31.0	248.0	42.3632	278561440	-71.3953	4456.0	1.0	3022	4.203390	292659	2020	4.720143
99998	19.0	22	3446	3500.0	26.0	280.0	42.8770	279630854	-72.2333	2.0	NaN	0	5.000000	59061	2021	4.617839
99999	22.0	265	48327	2500.0	29.0	197.0	42.6595	261052660	-83.4022	NaN	NaN	0	4.105263	50111	2020	4.456685

100000 rows × 16 columns

In [31]:

```python
# Even after removing columns with 50% missing values, the remaining variables also have many observations with missing values.
# We will impute these missing values using the Multivariate Imputation by Chained Equations (MICE) approach.

from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

df_mice = df_cleanNum.copy(deep=True)
mice_imputer = IterativeImputer()

df_mice.iloc[:,0:16] = mice_imputer.fit_transform(df_mice.iloc[:,0:16])
df_mice.iloc[:,0:16].isnull().sum()
```

Out[31]:

```
city_fuel_economy       0
daysonmarket            0
dealer_zip              0
engine_displacement     0
highway_fuel_economy    0
horsepower              0
latitude                0
listing_id              0
longitude               0
mileage                 0
owner_count             0
savings_amount          0
seller_rating           0
sp_id                   0
year                    0
log10_price             0
dtype: int64
```

In [32]:

```python
df_mice.head()
```

Out[32]:

	city_fuel_economy	daysonmarket	dealer_zip	engine_displacement	highway_fuel_economy	horsepower	latitude	listing_id	longitude	mileage	owner_count	savings_amount	seller_rating	sp_id	year	log10_price
0	17.720172	522	960	1300.0	31.164089	177.0	18.3988	237132766	-66.1582	7.000000	1.229669	0	2.80000	370599	2019	4.364382
1	19.243680	207	922	2000.0	30.554924	246.0	18.4439	265946296	-66.0785	8.000000	1.094793	0	3.00000	389227	2020	4.667453
2	17.000000	1233	969	2500.0	23.000000	305.0	18.3467	173473508	-66.1098	8846.303906	3.000000	0	1.64467	370467	2016	4.672052
3	19.424641	196	922	3000.0	29.590618	340.0	18.4439	266911050	-66.0785	11.000000	1.134808	0	3.00000	389227	2020	4.828853
4	19.313519	137	922	2000.0	30.572730	246.0	18.4439	270957414	-66.0785	7.000000	0.967691	0	3.00000	389227	2020	4.689131

In [33]:

```
#dropping irrelevant numerical column
df_mice = df_mice.drop(['listing_id','sp_id'], axis = 1)
df_mice
```

Out[33]:

	city_fuel_economy	daysonmarket	dealer_zip	engine_displacement	highway_fuel_economy	horsepower	latitude	longitude	mileage	owner_count	savings_amount	seller_rating	year	log10_price
0	17.720172	522	960	1300.0	31.164089	177.0	18.3988	-66.1582	7.000000	1.229669	0	2.800000	2019	4.364382
1	19.243680	207	922	2000.0	30.554924	246.0	18.4439	-66.0785	8.000000	1.094793	0	3.000000	2020	4.667453
2	17.000000	1233	969	2500.0	23.000000	305.0	18.3467	-66.1098	8846.303906	3.000000	0	1.644670	2016	4.672052
3	19.424641	196	922	3000.0	29.590618	340.0	18.4439	-66.0785	11.000000	1.134808	0	3.000000	2020	4.828853
4	19.313519	137	922	2000.0	30.572730	246.0	18.4439	-66.0785	7.000000	0.967691	0	3.000000	2020	4.689131
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99995	16.000000	55	49091	6200.0	22.000000	420.0	41.7981	-85.4287	18038.217898	0.862026	0	5.000000	2020	4.703962
99996	19.000000	187	2920	2000.0	25.000000	240.0	41.7982	-71.4481	67989.000000	1.000000	1764	4.000000	2016	4.184691
99997	24.000000	34	1776	2000.0	31.000000	248.0	42.3632	-71.3953	4456.000000	1.000000	3022	4.203390	2020	4.720143
99998	19.000000	22	3446	3500.0	26.000000	280.0	42.8770	-72.2333	2.000000	0.622905	0	5.000000	2021	4.617839
99999	22.000000	265	48327	2500.0	29.000000	197.0	42.6595	-83.4022	-888.275552	1.000794	0	4.105263	2020	4.456685

100000 rows × 14 columns

In [34]:

```
#do pair-wise correlation between the variables. Many of the variables seem to be highly correlated therefore there is a need to conduct PCA to reduce
# the dimensionality of the dataset. For example, variables 'longitude' and 'dealer_zip' have a correlation of highly negative correlation of 98% therefore including both these variables
# as it is is redundant our analysis

x = df_mice.drop(['log10_price'], axis = 1)
corr = x.corr()
corr.style.background_gradient(cmap='coolwarm')
```

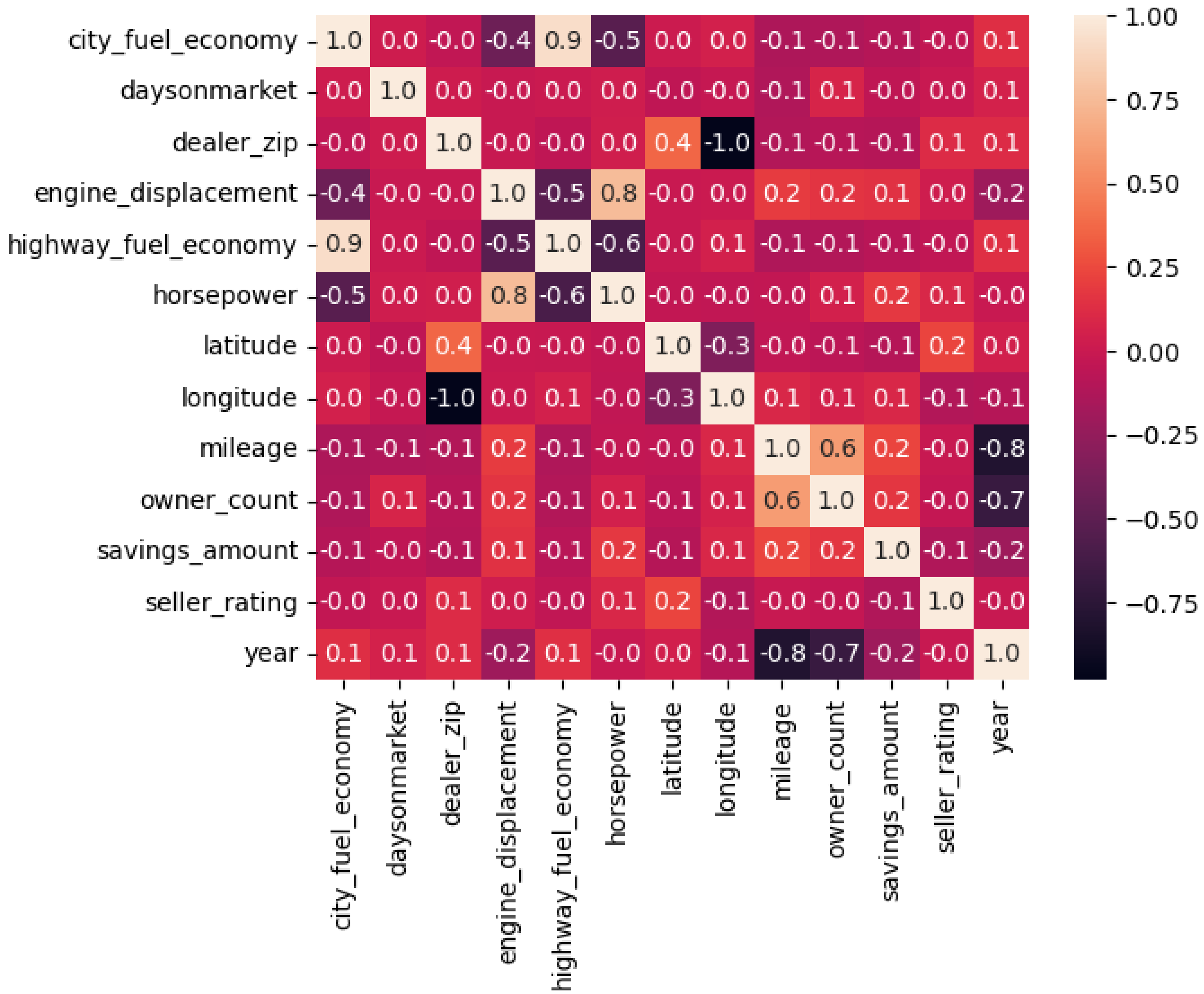
Out[34]:

	city_fuel_economy	daysonmarket	dealer_zip	engine_displacement	highway_fuel_economy	horsepower	latitude	longitude	mileage	owner_count	savings_amount	seller_rating	year
city_fuel_economy	1.000000	0.020887	-0.043245	-0.430908	0.927919	-0.524845	0.015134	0.049500	-0.134692	-0.122924	-0.095949	-0.036044	0.127261
daysonmarket	0.020887	1.000000	0.020473	-0.012478	0.002952	0.029877	-0.045088	-0.017996	-0.120289	0.077727	-0.046967	0.001331	0.058476
dealer_zip	-0.043245	0.020473	1.000000	-0.009597	-0.046298	0.037699	0.365235	-0.980085	-0.110616	-0.084715	-0.098251	0.112400	0.112019
engine_displacement	-0.430908	-0.012478	-0.009597	1.000000	-0.520661	0.753661	-0.000099	0.006606	0.189029	0.161303	0.145596	0.028964	-0.199483
highway_fuel_economy	0.927919	0.002952	-0.046298	-0.520661	1.000000	-0.595318	-0.007215	0.052268	-0.128849	-0.112280	-0.082562	-0.041858	0.139383
horsepower	-0.524845	0.029877	0.037699	0.753661	-0.595318	1.000000	-0.035482	-0.043208	-0.028937	0.054987	0.173728	0.093318	-0.002152
latitude	0.015134	-0.045088	0.365235	-0.000099	-0.007215	-0.035482	1.000000	-0.339016	-0.036790	-0.061740	-0.093484	0.231367	0.046121
longitude	0.049500	-0.017996	-0.980085	0.006606	0.052268	-0.043208	-0.339016	1.000000	0.088780	0.063596	0.088780	-0.107569	-0.092807
mileage	-0.134692	-0.120289	-0.110616	0.189029	-0.128849	-0.028937	-0.036790	0.088780	1.000000	0.604561	0.227873	-0.013067	-0.805111
owner_count	-0.122924	0.077727	-0.084715	0.161303	-0.112280	0.054987	-0.061740	0.063596	0.604561	1.000000	0.170565	-0.022750	-0.681418
savings_amount	-0.095949	-0.046967	-0.098251	0.145596	-0.082562	0.173728	-0.093484	0.088780	0.227873	0.170565	1.000000	-0.117730	-0.202366
seller_rating	-0.036044	0.001331	0.112400	0.028964	-0.041858	0.093318	0.231367	-0.107569	-0.013067	-0.022750	-0.117730	1.000000	-0.008549
year	0.127261	0.058476	0.112019	-0.199483	0.139383	-0.002152	0.046121	-0.092807	-0.805111	-0.681418	-0.202366	-0.008549	1.000000

In [35]:

```
#create heatmap using seaborn package

import seaborn as sns
x = df_mice.drop(['log10_price'], axis = 1)
ax = sns.heatmap(x.corr(), annot=True, fmt=".1f")
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [36]:

```
pos_threshold = 0.6
neg_threshold = -0.6

def high_poscor_function(df):
    cor = df.corr()
    corrm = np.corrcoef(df.transpose())
    corr = corrm - np.diagflat(corrm.diagonal())
    print("max corr:",corr.max(), ", min corr: ", corr.min())
    c1 = cor.stack().sort_values(ascending=False).drop_duplicates()
    high_cor = c1[c1.values!=1]
    thresh = pos_threshold
    display(high_cor[high_cor>thresh])

def high_negcor_function(df):
    cor = df.corr()
    corrm = np.corrcoef(df.transpose())
    corr = corrm - np.diagflat(corrm.diagonal())
    print("max corr:",corr.max(), ", min corr: ", corr.min())
    c1 = cor.stack().sort_values(ascending=False).drop_duplicates()
    high_cor = c1[c1.values!=1]
    thresh = neg_threshold
    display(high_cor[high_cor<thresh])
```

In [37]:

```
print('highly positively correlated variables are:', high_poscor_function(df_mice.drop('log10_price',axis = 1)))
print('highly negatively correlated variables are:', high_negcor_function(df_mice.drop('log10_price',axis = 1)))
```

```
max corr: 0.9279194843034885 , min corr:  -0.9800852281203005
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
highway_fuel_economy  city_fuel_economy      0.927919
horsepower            engine_displacement    0.753661
owner_count           mileage                0.604561
dtype: float64
highly positively correlated variables are: None
max corr: 0.9279194843034885 , min corr:  -0.9800852281203005
year       owner_count   -0.681418
mileage    year          -0.805111
longitude  dealer_zip    -0.980085
dtype: float64
highly negatively correlated variables are: None
```

Feature Engineering

In [38]:
```python
# Taking variables that are highly correlated and running PCA on them to reduce dimensionality
features = df_mice[['highway_fuel_economy','city_fuel_economy','horsepower','engine_displacement','owner_count','mileage','year','longitude','dealer_zip']]
features = features.columns
features
```

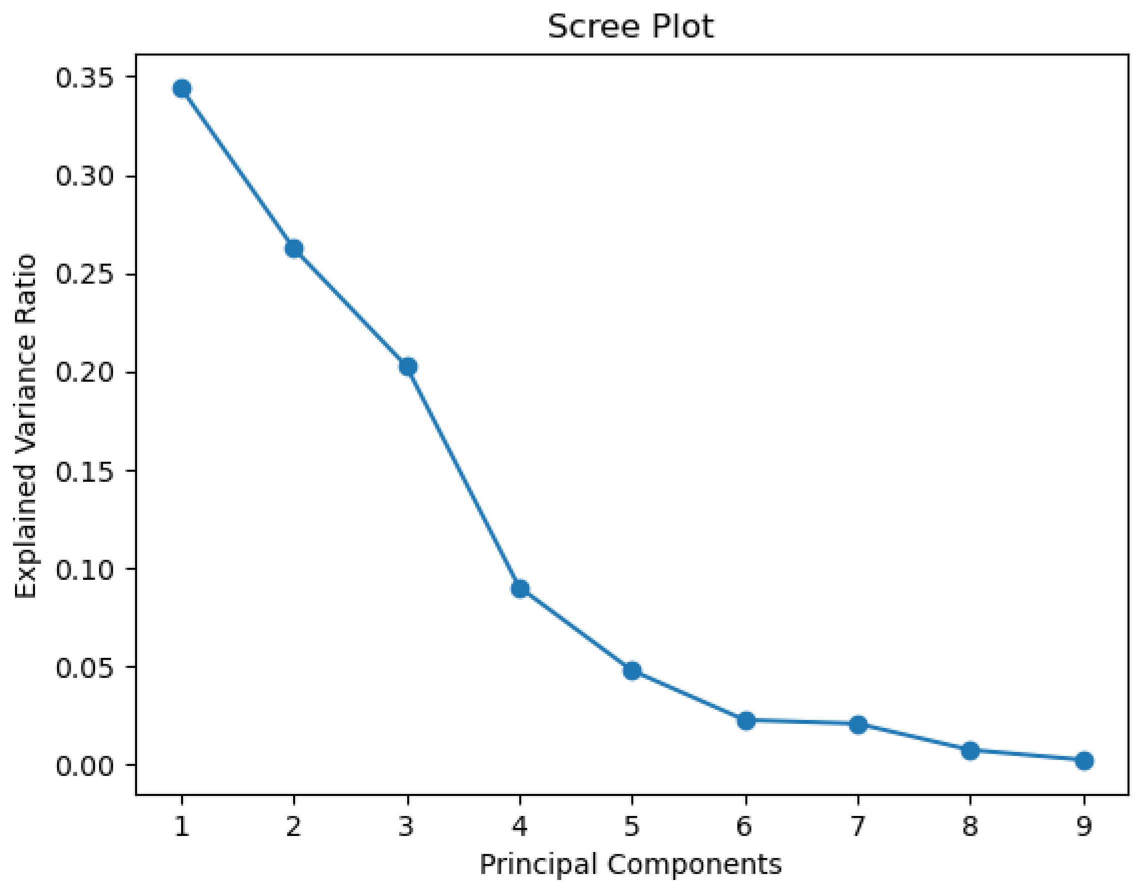
Out[38]:
```
Index(['highway_fuel_economy', 'city_fuel_economy', 'horsepower',
       'engine_displacement', 'owner_count', 'mileage', 'year', 'longitude',
       'dealer_zip'],
      dtype='object')
```

In [39]:
```python
from sklearn.preprocessing import StandardScaler

# Separating out the features
x = df_mice.loc[:, features].values

# Separating out the target
y = df_mice.loc[:,['log10_price']].values

# Standardizing the features
x = StandardScaler().fit_transform(x)
```

In [40]:
```python
from sklearn.decomposition import PCA

pca = PCA()

principalComponents = pca.fit_transform(x)
principalComponents
```

Out[40]:
```
array([[ 9.90330600e-01,  1.06735820e+00, -1.87905834e+00, ...,
         1.13741011e-01, -5.50363319e-01, -9.63386253e-01],
       [ 5.90952602e-01,  6.05824252e-01, -2.21903352e+00, ...,
        -8.68653233e-02, -3.93383422e-01, -9.78114847e-01],
       [-1.68263466e+00,  1.60916724e+00, -1.44851718e+00, ...,
         1.73412379e-01,  2.27863177e-01, -1.00480977e+00],
       ...,
       [ 9.39426652e-01,  2.19910495e-01, -1.31721491e+00, ...,
        -1.30220311e-01,  5.25521915e-03, -6.44742267e-02],
       [-1.74568148e-01, -8.81771454e-01, -1.74264720e+00, ...,
        -5.24462853e-02, -2.22197119e-02,  2.60832522e-02],
       [ 8.82228483e-01, -2.17264590e+00,  2.23838679e+00, ...,
         1.14261084e-01,  1.99862067e-03, -2.03403351e-02]])
```

In [41]:
```python
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
         pca.explained_variance_ratio_, marker='o')
plt.title('Scree Plot')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.show()
```



In [42]:
```
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = explained_variance_ratio.cumsum()
singular_values = pca.singular_values_

summary_df = pd.DataFrame({
    'PC': range(1, len(explained_variance_ratio) + 1),
    'Explained Variance Ratio': explained_variance_ratio,
    'Cumulative Variance Ratio': cumulative_variance_ratio,
    'Singular Values': singular_values
})

print(summary_df)
```

```
   PC  Explained Variance Ratio  Cumulative Variance Ratio  Singular Values
0   1                  0.344451                   0.344451       556.781602
1   2                  0.262635                   0.607086       486.180442
2   3                  0.202647                   0.809732       427.062096
3   4                  0.090125                   0.899857       284.801886
4   5                  0.047687                   0.947544       207.166829
5   6                  0.022480                   0.970024       142.238311
6   7                  0.020572                   0.990596       136.069441
7   8                  0.007231                   0.997827        80.672801
8   9                  0.002173                   1.000000        44.225007
```

In [43]:
```
pca_fnl = PCA(n_components=4)
pca_fnl.fit(x)

x_org = pca_fnl.fit_transform(x)
x_org
```

Out[43]:
```
array([[ 0.9903306 ,  1.0673582 , -1.87905834, -1.39356099],
       [ 0.5909526 ,  0.60582425, -2.21903352, -0.61855636],
       [-1.68263466,  1.60916724, -1.44851718, -0.56109426],
       ...,
       [ 0.93942665,  0.2199105 , -1.31721491, -0.19667717],
       [-0.17456815, -0.88177145, -1.7426472 ,  0.02545762],
       [ 0.88222848, -2.1726459 ,  2.23838679, -0.29882646]])
```

In [44]:
```python
# Principal components correlation coefficients
loadings = pca_fnl.components_

# Number of features before PCA
n_features = pca_fnl.n_features_in_

# Feature names before PCA
feature_names = features

# PC names
pc_list = [f'PC{i}' for i in list(range(1, n_features + 1))]

# Match PC names to loadings
pc_loadings = dict(zip(pc_list, loadings))

# Matrix of corr coefs between feature names and PCs
loadings_df = pd.DataFrame.from_dict(pc_loadings)
loadings_df['features'] = features
loadings_df = loadings_df.set_index('features')
loadings_df
```

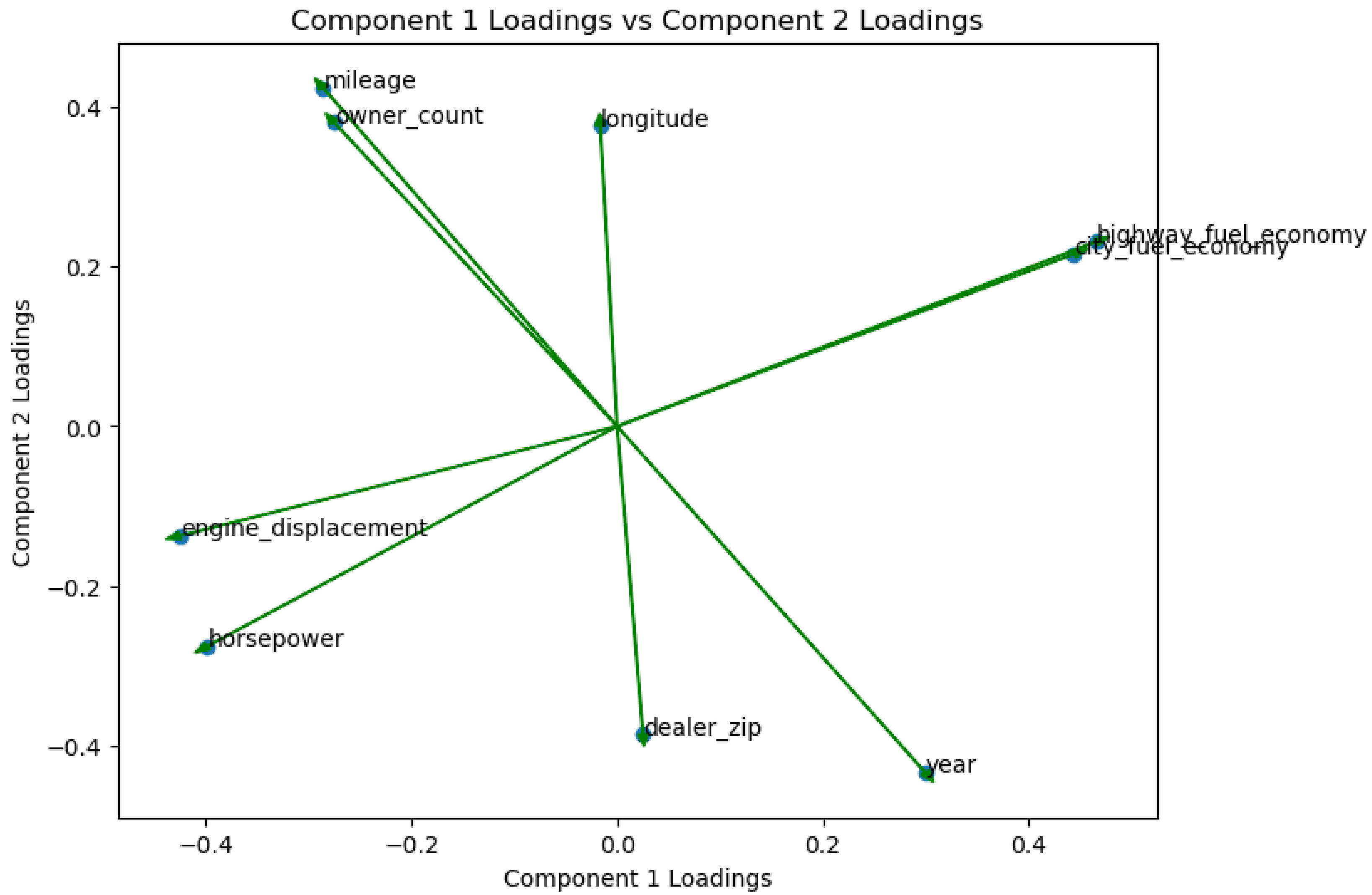
Out[44]:

features	PC1	PC2	PC3	PC4
highway_fuel_economy	0.465391	0.231496	0.110515	0.419447
city_fuel_economy	0.443897	0.216125	0.099114	0.532225
horsepower	-0.398064	-0.275613	-0.176944	0.442052
engine_displacement	-0.424726	-0.137108	-0.110590	0.582958
owner_count	-0.274748	0.380711	0.267693	0.053978
mileage	-0.285613	0.424469	0.277149	-0.029163
year	0.299256	-0.433915	-0.283444	-0.008753
longitude	-0.016465	0.377297	-0.597998	-0.028622
dealer_zip	0.025168	-0.386249	0.589924	0.029864

In [45]:
```python
loadings_component1 = pca_fnl.components_[0]
loadings_component2 = pca_fnl.components_[1]

plt.figure(figsize=(8, 6))
plt.scatter(loadings_component1, loadings_component2, marker='o')
plt.title('Component 1 Loadings vs Component 2 Loadings')
plt.xlabel('Component 1 Loadings')
plt.ylabel('Component 2 Loadings')

for i, feature in enumerate(features):
    plt.annotate(feature, (loadings_component1[i], loadings_component2[i]))
    plt.arrow(
        0, 0, # coordinates of arrow base
        loadings_component1[i], # length of the arrow along x
                                  ength of the arrow along y
        color='g',
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
        head_width=0.01
        )
plt.show()
```



PC1 - high_fuel_economy and city_fuel_economy --> fuel economy PC2 - mileage, year and owner_count --> mileage PC3 - longitude and dealer_zip --> dealer location PC4 - engine_displacement and horsepower --> engine power

In [46]:
```
#converting principal components to original scale -- need help here!

scaler = StandardScaler()
scaler.fit(x_org)
# X_orig = np.dot(x_org, pca_fnl.components_) don't need this
```

Out[46]:
```
▾ StandardScaler
StandardScaler()
```

In [47]:
```
scaler = StandardScaler()
scaler.fit(x_org)

# Perform the inverse transformation
X_orig_backscaled = scaler.inverse_transform(x_org)
```

In [48]:
```
X_orig_backscaled
```

Out[48]:
```
array([[ 1.74367313,  1.64099657, -2.53764748, -1.25507258],
       [ 1.04048908,  0.93141696, -2.99678019, -0.55708586],
       [-2.96261152,  2.47399414, -1.95620641, -0.5053342 ],
       ...,
       [ 1.65404665,  0.33809865, -1.77888415, -0.17713191],
       [-0.30736179, -1.35566854, -2.35342575,  0.02292771],
       [ 1.55333795, -3.34030739,  3.02291658, -0.26912988]])
```

In [49]:
```
df_pca = pd.DataFrame(data = X_orig_backscaled
             , columns = ['fuel_economy', 'mileage','dealer_location','engine_power'])
df_pca
```

Out[49]:

	fuel_economy	mileage	dealer_location	engine_power
0	1.743673	1.640997	-2.537647	-1.255073
1	1.040489	0.931417	-2.996780	-0.557086
2	-2.962612	2.473994	-1.956206	-0.505334
3	-0.515961	0.271485	-3.387703	0.300200
4	1.139372	0.827917	-3.061594	-0.560325
...	...	...	...	...
99995	-4.354919	-5.944582	2.191628	1.661834
99996	-0.887101	1.442887	-0.955877	-0.926741
99997	1.654047	0.338099	-1.778884	-0.177132
99998	-0.307362	-1.355669	-2.353426	0.022928
99999	1.553338	-3.340307	3.022917	-0.269130

100000 rows × 4 columns

In [50]:
```
# Combining the final list of variables: get the original categorical and numerical variables and combine them with the principal components:

pd.set_option('display.max_columns', 1000)
df_num = df_mice.drop(features,axis = 1)

df_final = pd.concat([df_num, df_catg, df_pca], axis=1)
df_final.shape
df_final.head()
```

Out[50]:

	daysonmarket	latitude	savings_amount	seller_rating	log10_price	back_legroom	body_type	city	engine_cylinders	engine_type	exterior_color	fleet	frame_damaged	franchise_make	front_legroom	fuel_tank_volume	fu
0	522	18.3988	0	2.80000	4.364382	35.1 in	SUV / Crossover	Bayamon	I4	I4	Solar Yellow	Missing	Missing	Jeep	41.2 in	12.7 gal	
1	207	18.4439	0	3.00000	4.667453	38.1 in	SUV / Crossover	San Juan	I4	I4	Narvik Black	Missing	Missing	Land Rover	39.1 in	17.7 gal	
2	1233	18.3467	0	1.64467	4.672052	35.4 in	Sedan	Guaynabo	H4	H4	Missing	False	False	FIAT	43.3 in	15.9 gal	
3	196	18.4439	0	3.00000	4.828853	37.6 in	SUV / Crossover	San Juan	V6	V6	Eiger Gray	Missing	Missing	Land Rover	39 in	23.5 gal	
4	137	18.4439	0	3.00000	4.689131	38.1 in	SUV / Crossover	San Juan	I4	I4	Narvik Black	Missing	Missing	Land Rover	39.1 in	17.7 gal	

In [51]:

```
#removing variables based on industry knowledge
df_subset = df_final.drop(['latitude','power','engine_cylinders','wheel_system_display'], axis = 1)
```

In [52]:

```
# removing characters from potential numerical variables

df_subset['wheelbase'] = df_subset['wheelbase'].str.split().str[0]
df_subset['back_legroom'] = df_subset['back_legroom'].str.split().str[0]
df_subset['front_legroom'] = df_subset['front_legroom'].str.split().str[0]
df_subset['fuel_tank_volume'] = df_subset['fuel_tank_volume'].str.split().str[0]
df_subset['height'] = df_subset['height'].str.split().str[0]
df_subset['length'] = df_subset['length'].str.split().str[0]
df_subset['width'] = df_subset['width'].str.split().str[0]

cols = ['wheelbase','back_legroom','front_legroom','fuel_tank_volume','height','length','width']

df_subset[cols] = df_subset[cols].replace('Missing', 0).replace('--', 0).apply(pd.to_numeric)
```

In [53]:

```
def convert(value):
    if value == True:
        return 1
    elif value == False:
        return 0
    else:
        return 999

df_subset['frame_damaged'] = df_subset['frame_damaged'].apply(lambda x: convert(x))
df_subset['fleet'] = df_subset['fleet'].apply(lambda x: convert(x))
df_subset['has_accidents'] = df_subset['has_accidents'].apply(lambda x: convert(x))
df_subset['isCab'] = df_subset['isCab'].apply(lambda x: convert(x))
df_subset['theft_title'] = df_subset['theft_title'].apply(lambda x: convert(x))
df_subset['salvage'] = df_subset['salvage'].apply(lambda x: convert(x))

df_subset
```

Out[53]:

	daysonmarket	savings_amount	seller_rating	log10_price	back_legroom	body_type	city	engine_type	exterior_color	fleet	frame_damaged	franchise_make	front_legroom	fuel_tank_volume	fuel_type	has_accidents
0	522	0	2.800000	4.364382	35.1	SUV / Crossover	Bayamon	I4	Solar Yellow	999	999	Jeep	41.2	12.7	Gasoline	999
1	207	0	3.000000	4.667453	38.1	SUV / Crossover	San Juan	I4	Narvik Black	999	999	Land Rover	39.1	17.7	Gasoline	999
2	1233	0	1.644670	4.672052	35.4	Sedan	Guaynabo	H4	Missing	0	0	FIAT	43.3	15.9	Gasoline	0
3	196	0	3.000000	4.828853	37.6	SUV / Crossover	San Juan	V6	Eiger Gray	999	999	Land Rover	39.0	23.5	Gasoline	999
4	137	0	3.000000	4.689131	38.1	SUV / Crossover	San Juan	I4	Narvik Black	999	999	Land Rover	39.1	17.7	Gasoline	999
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99995	55	0	5.000000	4.703962	43.4	Pickup Truck	Sturgis	V8	Northsky Blue Metallic	999	999	Chevrolet	44.5	24.0	Gasoline	999
99996	187	1764	4.000000	4.184691	39.4	SUV / Crossover	Cranston	I4	Silver	0	0	Missing	44.1	18.8	Gasoline	0
99997	34	3022	4.203390	4.720143	36.5	Sedan	Sudbury	I4	Mediterranean Blue Metallic	0	0	BMW	41.4	18.0	Gasoline	0
99998	22	0	5.000000	4.617839	38.4	SUV / Crossover	Swanzey	V6	Gray	999	999	Honda	40.9	19.5	Gasoline	999
99999	265	0	4.105263	4.456685	37.5	SUV / Crossover	Waterford	I4	Ebony Twilight Metallic	999	999	Buick	40.9	17.3	Gasoline	999

100000 rows × 38 columns

In [54]:
```
# file_path = 'data.csv'
# df_subset.to_csv(file_path, index=False)
# print(f"Data has been exported to {file_path}.")
```

In [55]:
```
# from google.colab import files

# files.download(file_path)
```

Linear Models

In [56]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from patsy import dmatrices
from sklearn.metrics import mean_squared_error
from math import sqrt
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [57]:
```
#df = pd.read_csv('data.csv')
df_lm = df_subset.drop(columns=['interior_color', 'exterior_color', 'city', 'trim_name' , 'sp_name'])
df_lm.head()
```

Out[57]:

	daysonmarket	savings_amount	seller_rating	log10_price	back_legroom	body_type	engine_type	fleet	frame_damaged	franchise_make	front_legroom	fuel_tank_volume	fuel_type	has_accidents	height	isCab	length	listed_d
0	522	0	2.80000	4.364382	35.1	SUV / Crossover	I4	999	999	Jeep	41.2	12.7	Gasoline	999	66.5	999	166.6	2019-
1	207	0	3.00000	4.667453	38.1	SUV / Crossover	I4	999	999	Land Rover	39.1	17.7	Gasoline	999	68.0	999	181.0	2020-
2	1233	0	1.64467	4.672052	35.4	Sedan	H4	0	0	FIAT	43.3	15.9	Gasoline	0	58.1	0	180.9	2017-
3	196	0	3.00000	4.828853	37.6	SUV / Crossover	V6	999	999	Land Rover	39.0	23.5	Gasoline	999	73.0	999	195.1	2020-
4	137	0	3.00000	4.689131	38.1	SUV / Crossover	I4	999	999	Land Rover	39.1	17.7	Gasoline	999	68.0	999	181.0	2020-

In [58]:
```
# Convert log10_price to normal price
df_lm['price'] = np.power(10, df_lm['log10_price'])
df_lm.drop(columns=['log10_price'], inplace=True)

# Prevent dates from being one hot encoded: Date them number of days from 30 Sept 2020, which is the estimated day that th edata was generated.

# Convert original date column to date type
df_lm['listed_date'] = pd.to_datetime(df_lm['listed_date'], format='%Y-%m-%d')

# Calculate days from date
reference_date = pd.to_datetime('2020-09-30')
df_lm['days_listed'] = (-(df_lm['listed_date'] - reference_date)).dt.days

df_lm.drop(columns=['listed_date'], inplace=True)
```

In [59]:
```
# Choose variables
from sklearn.model_selection import train_test_split
formula = 'price ~ make_name + model_name + fuel_type + mileage + days_listed'  # Automatically handles 'x1' as categorical
y, X = dmatrices(formula, data=df_lm, return_type='dataframe')

# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=123)
```

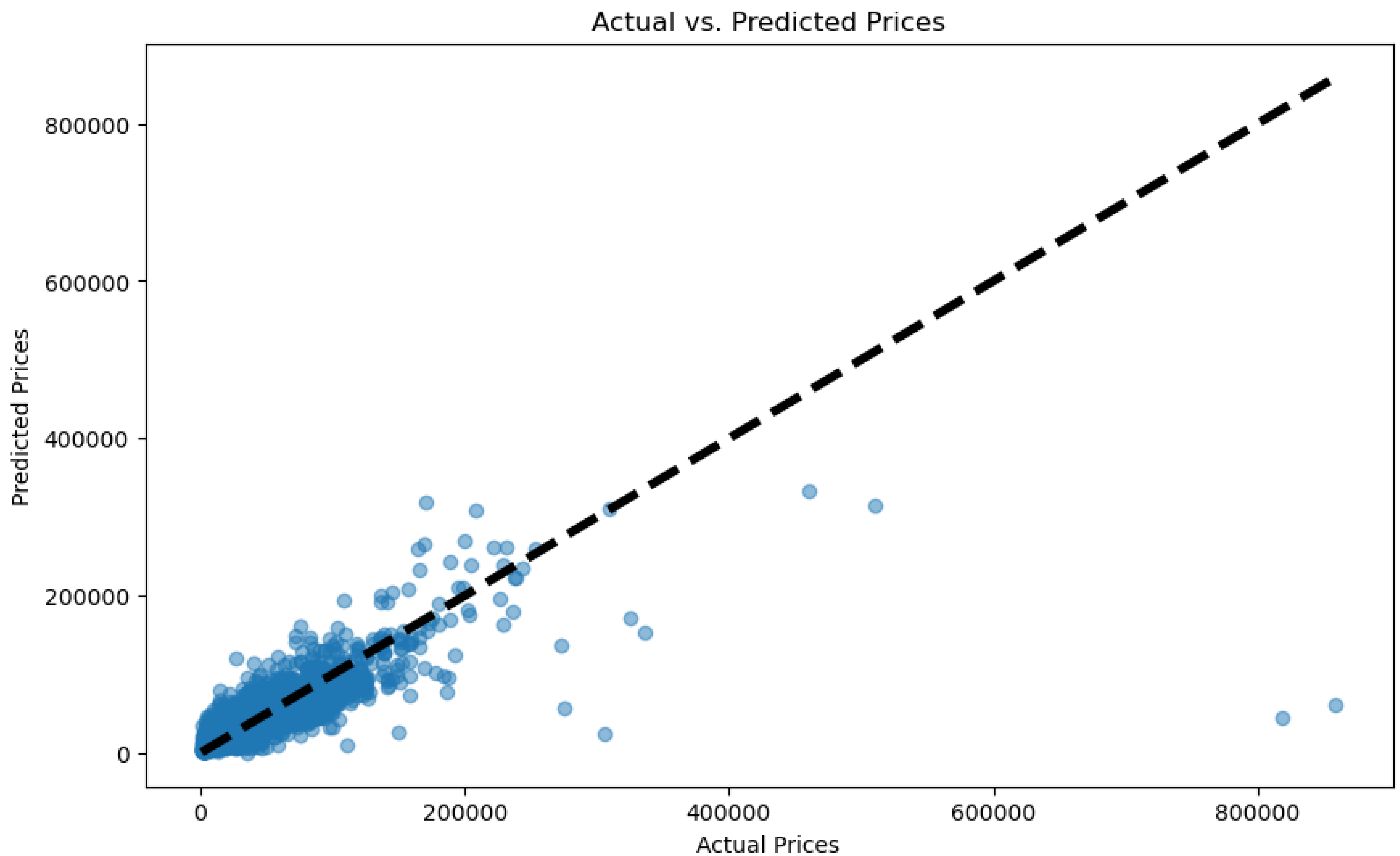
In [60]:
```
#GLM MODEL

TG_model = sm.GLM(y_train, X_train, family=sm.families.Gaussian(sm.families.links.log())).fit()
gy_pred = TG_model.predict(X_test)

# RMSE
rmse = sqrt(mean_squared_error(y_test, gy_pred))
print(f'RMSE: {rmse}')

# R2
r2 = r2_score(y_test, gy_pred)
print(f'R^2: {r2}')
```

```
/Users/foroughmofidi/anaconda3/lib/python3.11/site-packages/statsmodels/genmod/families/links.py:13: FutureWarning: The log link alias is deprecated. Use Log instead. The log link alias will b
e removed after the 0.15.0 release.
  warnings.warn(
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
RMSE: 11719.059092439848
R^2: 0.6778321592820891
```

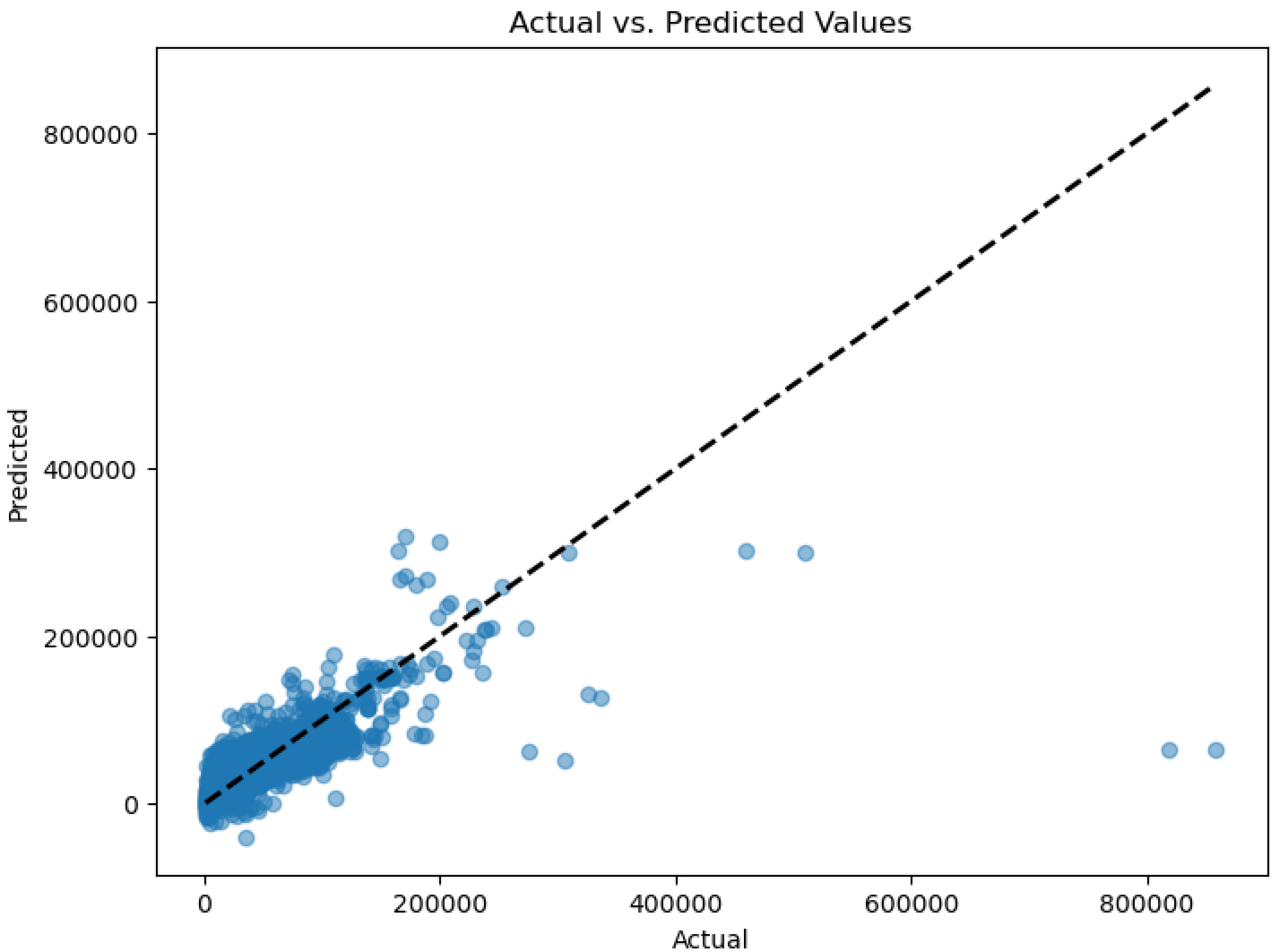
In [61]:
```python
# Visualise GLM Predictions

plt.figure(figsize=(10, 6))
plt.scatter(y_test, gy_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs. Predicted Prices')
plt.show()
```



In [62]:
```python
# OLS MODEL

TO_model = sm.OLS(y_train, X_train).fit()
y_pred = TO_model.predict(X_test)

# RMSE
rmse = sqrt(mean_squared_error(y_test, y_pred))
print(f'RMSE: {rmse}')

# R2
r2_score_test = r2_score(y_test, y_pred)
print(f'R^2 score for the test set: {r2_score_test}')
```

```
RMSE: 11806.488479409896
R^2 score for the test set: 0.6730071975640429
```

In [63]:
```python
# Visualise OLS Predictions

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)  # Diagonal line
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs. Predicted Values')
plt.show()
```



Explainable Boosting Machine (EBM)

In [101…
```python
import pandas as pd
import numpy as np

pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', 500)
```

In [102…
```python
#!pip install interpret-core dash-cytoscape
from interpret import set_visualize_provider
from interpret.provider import InlineProvider
set_visualize_provider(InlineProvider())
```

In [103
```python
import numpy as np
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score,accuracy_score,f1_score,confusion_matrix
from interpret.glassbox import ExplainableBoostingClassifier
from interpret.glassbox import ClassificationTree
from interpret.data import Marginal
from interpret.perf import RegressionPerf
from interpret.glassbox import ExplainableBoostingRegressor, LinearRegression, RegressionTree

from interpret import show
import matplotlib.pyplot as plt
import seaborn as sns
```

Read Data - From CSV File

In [138…

```
# Read from CSV File "data.csv"
df = pd.read_csv("data.csv")
df = df.drop(columns=['interior_color', 'exterior_color' , 'city' , 'trim_name' , 'sp_name'])

#df_ebm = df_subset
```

In [139…

```
# Count unique values in df
unique_categories_counts = {}

for column in df.columns:
    unique_categories_counts[column] = df[column].nunique()

print("Number of unique categories in each column:")
print(unique_categories_counts)
```

```
Number of unique categories in each column:
{'daysonmarket': 848, 'savings_amount': 5663, 'seller_rating': 1048, 'log10_price': 27707, 'back_legroom': 187, 'body_type': 10, 'engine_type': 30, 'fleet': 3, 'frame_damaged': 3, 'franchise_make': 41, 'front_legroom': 82, 'fuel_tank_volume': 160, 'fuel_type': 8, 'has_accidents': 3, 'height': 415, 'isCab': 3, 'length': 702, 'listed_date': 849, 'listing_color': 15, 'make_name': 59, 'maximum_seating': 13, 'model_name': 843, 'salvage': 3, 'theft_title': 3, 'transmission': 5, 'transmission_display': 36, 'wheel_system': 6, 'wheelbase': 400, 'width': 259, 'fuel_economy': 99988, 'mileage': 99988, 'dealer_location': 99988, 'engine_power': 99988}
```

Categorical Variables

In [140…

```
column_types = df.dtypes
print(column_types)
categorical_cols = ['body_type','engine_type', 'franchise_make' , 'fuel_type', 'listed_date', 'listing_color' , 'make_name',
                    'maximum_seating', 'model_name','transmission','transmission_display','wheel_system']
```

```
daysonmarket             float64
savings_amount           float64
seller_rating            float64
log10_price              float64
back_legroom             float64
body_type                 object
engine_type               object
fleet                      int64
frame_damaged              int64
franchise_make            object
front_legroom            float64
fuel_tank_volume         float64
fuel_type                 object
has_accidents              int64
height                   float64
isCab                      int64
length                   float64
listed_date               object
listing_color             object
make_name                 object
maximum_seating           object
model_name                object
salvage                    int64
theft_title                int64
transmission              object
transmission_display      object
wheel_system              object
wheelbase                float64
width                    float64
fuel_economy             float64
mileage                  float64
dealer_location          float64
engine_power             float64
dtype: object
```

Convert to dummies

In [109…
```
data_dummies = pd.get_dummies(df[categorical_cols], drop_first=True)
data_dummies = data_dummies.astype(int) # 0 and 1
data_dummies.head()
```

Out[109]:

	body_type_Coupe	body_type_Hatchback	body_type_Minivan	body_type_Pickup Truck	body_type_SUV / Crossover	body_type_Sedan	body_type_Van	body_type_Wagon	body_type_Missing	engine_type_H4 Hybrid	engine_type_H6	engine_type_I2	engine
0	0	0	0	0	1	0	0	0	0	0	0	0	
1	0	0	0	0	1	0	0	0	0	0	0	0	
2	0	0	0	0	0	1	0	0	0	0	0	0	
3	0	0	0	0	1	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	0	0	0	

Merge dummies with numerical cols¶

In [141…
```
noncategorical_cols = [col for col in df.columns if col not in categorical_cols]
noncategorical_cols
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Out[141]: ['daysonmarket',
 'savings_amount',
 'seller_rating',
 'log10_price',
 'back_legroom',
 'fleet',
 'frame_damaged',
 'front_legroom',
 'fuel_tank_volume',
 'has_accidents',
 'height',
 'isCab',
 'length',
 'salvage',
 'theft_title',
 'wheelbase',
 'width',
 'fuel_economy',
 'mileage',
 'dealer_location',
 'engine_power']
```

```
In [142… data_preprocess = pd.concat([df[noncategorical_cols], data_dummies], axis=1)
```

```
In [143… # Sample
sample_df = data_preprocess.sample(frac=0.5)
sample_df.shape
```

```
Out[143]: (50000, 1928)
```

```
In [144… # numeric Columns
numeric = df[noncategorical_cols]
```

Train , Test

```
In [145… X = sample_df.drop('log10_price', axis=1)  # Features: the entire df except target column
y = sample_df['log10_price']  # Target: just the column "log price"
```

```
In [146… X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

EBM Model

```
In [147… seed = 42
np.random.seed(seed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=seed)
```

```
In [148… %%time
ebm = ExplainableBoostingRegressor(random_state=seed, interactions=0)
```

```
CPU times: user 62 µs, sys: 165 µs, total: 227 µs
Wall time: 241 µs
```

```
In [149… %%time
ebm.fit(X_train, y_train)
```

```
CPU times: user 8.44 s, sys: 200 ms, total: 8.64 s
Wall time: 13min 12s
```

Out[149]:
```
▾ ExplainableBoostingRegressor
ExplainableBoostingRegressor(interactions=0)
```

Global Explanation¶

In [150…
```
import interpret
from interpret import show
from interpret import preserve

local_explanations = ebm.explain_local(X_test, y_test, name='EBM')

# Use preserve to save the visualization
preserve(local_explanations, file_name="local_explanations.html")
```

In [151…
```
import os

current_working_directory = os.getcwd()
#print("Current Working Directory:", current_working_directory)
```

In [152…
```
show(ebm.explain_global())
```

Local Explanations

In [153…
```
local_explanations = ebm.explain_local(X_test, y_test, name='EBM')
show(local_explanations)
```

Evaluate EBM performance

In [154…
```
column_names = sample_df.columns
column_names
```

Out[154]:
```
Index(['daysonmarket', 'savings_amount', 'seller_rating', 'log10_price',
       'back_legroom', 'fleet', 'frame_damaged', 'front_legroom',
       'fuel_tank_volume', 'has_accidents',
       ...
       'transmission_display_9-Speed Automatic Overdrive',
       'transmission_display_Automatic',
       'transmission_display_Continuously Variable Transmission',
       'transmission_display_Manual', 'transmission_display_Missing',
       'wheel_system_4X2', 'wheel_system_AWD', 'wheel_system_FWD',
       'wheel_system_RWD', 'wheel_system_Missing'],
      dtype='object', length=1928)
```

In [155…
```
column_names = [col for col in column_names if col != 'log10_price']
```

In [156…
```
ebm_perf = RegressionPerf(ebm, column_names).explain_perf(X_test, y_test, name='EBM')
show(ebm_perf)
```

Other Explainable Models

In [157…
```python
lr = LinearRegression(column_names)
lr.fit(X_train, y_train)

rt = RegressionTree(column_names, random_state=seed)
rt.fit(X_train, y_train)
```

Out[157]:
```
<interpret.glassbox._decisiontree.RegressionTree at 0x4a4ff6050>
```

Compare performance using the Dashboard

In [158…
```python
lr_perf = RegressionPerf(lr, column_names).explain_perf(X_test, y_test, name='Linear Regression')
show(lr_perf)
```

In [159…
```python
rt_perf = RegressionPerf(rt, column_names).explain_perf(X_test, y_test, name='Regression Tree')
show(rt_perf)
```

Glassbox

In [160…
```python
lr_global = lr.explain_global(name='Linear Regression')
show(lr_global)
```

In [175…
```python
rt_global = rt.explain_global(name='Regression Tree')
show(rt_global)
```

Propensity Score Matching

In [176…
```python
#pip install causalinference
```

In [177…
```python
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from causalinference import CausalModel
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
```

In [167…
```python
#df = pd.read_csv("data.csv")
#df = df.drop(columns=['interior_color', 'exterior_color' , 'city' , 'trim_name' , 'sp_name'])
#df_psm = df_subset
```

In [178…
```python
df['price'] = np.power(10, df['log10_price'])
df.drop(columns=['log10_price'], inplace=True)
```

```python
d vehicles
hybrid_types = [
```

```
    'V6 Hybrid',
    'I4 Hybrid',
    'H4 Hybrid',
    'V8 Hybrid'
]

# Set a flag indicating whether each vehicle is a hybrid (1) or not (0)
df['is_hybrid'] = df_psm['engine_type'].apply(lambda x: 1 if x in hybrid_types else 0)
```

In [180…

```
df.head()
```

Out[180]:

	daysonmarket	savings_amount	seller_rating	back_legroom	body_type	engine_type	fleet	frame_damaged	franchise_make	front_legroom	fuel_tank_volume	fuel_type	has_accidents	height	isCab	length	listed_date	listing_
0	522.0	0.0	2.80000	35.1	SUV / Crossover	I4	999	999	Jeep	41.2	12.7	Gasoline	999	66.5	999	166.6	2019-04-06	YEl
1	207.0	0.0	3.00000	38.1	SUV / Crossover	I4	999	999	Land Rover	39.1	17.7	Gasoline	999	68.0	999	181.0	2020-02-15	B
2	1233.0	0.0	1.64467	35.4	Sedan	H4	0	0	FIAT	43.3	15.9	Gasoline	0	58.1	0	180.9	2017-04-25	UNKN
3	196.0	0.0	3.00000	37.6	SUV / Crossover	V6	999	999	Land Rover	39.0	23.5	Gasoline	999	73.0	999	195.1	2020-02-26	
4	137.0	0.0	3.00000	38.1	SUV / Crossover	I4	999	999	Land Rover	39.1	17.7	Gasoline	999	68.0	999	181.0	2020-04-25	B

In [185…

```
df.info()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 34 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   daysonmarket          100000 non-null  float64
 1   savings_amount        100000 non-null  float64
 2   seller_rating         100000 non-null  float64
 3   back_legroom          100000 non-null  float64
 4   body_type             100000 non-null  object 
 5   engine_type           100000 non-null  object 
 6   fleet                 100000 non-null  int64  
 7   frame_damaged         100000 non-null  int64  
 8   franchise_make        100000 non-null  object 
 9   front_legroom         100000 non-null  float64
 10  fuel_tank_volume      100000 non-null  float64
 11  fuel_type             100000 non-null  object 
 12  has_accidents         100000 non-null  int64  
 13  height                100000 non-null  float64
 14  isCab                 100000 non-null  int64  
 15  length                100000 non-null  float64
 16  listed_date           100000 non-null  object 
 17  listing_color         100000 non-null  object 
 18  make_name             100000 non-null  object 
 19  maximum_seating       100000 non-null  object 
 20  model_name            100000 non-null  object 
 21  salvage               100000 non-null  int64  
 22  theft_title           100000 non-null  int64  
 23  transmission          100000 non-null  object 
 24  transmission_display  100000 non-null  object 
 25  wheel_system          100000 non-null  object 
 26  wheelbase             100000 non-null  float64
 27  width                 100000 non-null  float64
 28  fuel_economy          100000 non-null  float64
 29  mileage               100000 non-null  float64
 30  dealer_location       100000 non-null  float64
 31  engine_power          100000 non-null  float64
 32  price                 100000 non-null  float64
 33  is_hybrid             100000 non-null  int64  
dtypes: float64(15), int64(7), object(12)
memory usage: 25.9+ MB
```

In [186…

```python
# Specify the names of categorical and numerical columns
categorical_cols = df.select_dtypes(include=['object']).columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns.drop(['price', 'is_hybrid'])

# Create a ColumnTransformer for encoding categorical variables and scaling numerical variables
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
        ('cat', OneHotEncoder(), categorical_cols)
    ]
)

# Create a pipeline
pipeline = make_pipeline(
    preprocessor,
    LogisticRegression()
)

# Define features and target variable
X = df.drop(columns=['price', 'is_hybrid'])
y = df['is_hybrid']
```

```python
# Train the model using the pipeline
pipeline.fit(X, y)

# Calculate propensity scores
df['propensity_score'] = pipeline.predict_proba(X)[:, 1]
```

In [187…

```python
# Initialize the CausalModel
causal = CausalModel(
    Y=df['price'].values,
    D=df['is_hybrid'].values,
    X=df['propensity_score'].values
)

# Perform matching
causal.est_via_matching()

# Display the results
print(causal.estimates)
```

```
Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE -24881.877  12082.623     -2.059      0.039 -48563.818  -1199.937
           ATC -25049.739  12303.722     -2.036      0.042 -49165.035   -934.443
           ATT -15971.235  21042.825     -0.759      0.448 -57215.172  25272.702
```

The results indicate that being a hybrid vehicle (HV) has a statistically significant negative impact on price (average price difference = $25,000). However, based on the results from the EBM analysis, it is understood that in the US market, vehicles with conflicting characteristics of high fuel efficiency and high engine power are advantageous. Typically, both HV and EV (Electric Vehicles) would meet these conditions, but given the period of data collection, only HVs were considered, leading to the observed negative impact. It is hypothesized that while HVs show a negative impact on the market overall, there might be a strong premium on specific brands.

In [188…

```python
# Column name storing the brand names
brand_col = 'make_name'

# Loop through each brand
for brand in df[brand_col].unique():
    # Extract data for the current brand
    df_brand = df[df[brand_col] == brand].copy()

    # Check if both hybrid and non-hybrid vehicles exist
    if df_brand['is_hybrid'].nunique() < 2:
        continue  # Skip if both classes do not exist

    # Specify the names of categorical and numerical columns
    # Exclude the brand name column
    categorical_cols = df_brand.select_dtypes(include=['object']).columns.drop([brand_col])
    numeric_cols = df_brand.select_dtypes(include=['int64', 'float64']).columns.drop(['price', 'is_hybrid'])

    # Create a ColumnTransformer for encoding categorical variables and scaling numerical variables
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), numeric_cols),
            ('cat', OneHotEncoder(), categorical_cols)
        ]
    )

    # Create a pipeline
    pipeline = make_pipeline(
        preprocessor,
        LogisticRegression()
    )
```

```python
    # Define features and target variable
    # Exclude the brand name column as well
    X = df_brand.drop(columns=['price', 'is_hybrid', brand_col])
    y = df_brand['is_hybrid']

    # Train the model using the pipeline
    pipeline.fit(X, y)

    # Calculate propensity scores
    df.loc[df[brand_col] == brand, 'propensity_score'] = pipeline.predict_proba(X)[:, 1]
```

In [189…

```python
# Column name storing the brand names
brand_col = 'make_name'

# Loop through each brand
for brand in df[brand_col].unique():
    # Extract data for the current brand
    df_brand = df[df[brand_col] == brand].copy()

    # Check if both hybrid and non-hybrid vehicles exist
    if df_brand['is_hybrid'].nunique() < 2:
        continue  # Skip if both classes do not exist

    # Check if propensity scores are calculated and not null
    if 'propensity_score' not in df_brand.columns or df_brand['propensity_score'].isnull().any():
        print(f"Skipping {brand} due to missing propensity scores.")
        continue

    try:
        # Initialize the CausalModel
        causal = CausalModel(
            Y=df_brand['price'].values,
            D=df_brand['is_hybrid'].values,
            X=df_brand[['propensity_score']].values
        )

        # Perform matching
        causal.est_via_matching()

        # Display the results
        print(f"Brand: {brand}")
        print(causal.estimates)
        print("\n")
    except ValueError as e:
        print(f"Skipping {brand} due to an error: {e}")
        continue
```

```
Brand: Subaru

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE -11136.403   2090.687     -5.327      0.000 -15234.149  -7038.657
           ATC -11167.406   2093.186     -5.335      0.000 -15270.051  -7064.760
           ATT   9096.200   9202.850      0.988      0.323  -8941.385  27133.785



Brand: Hyundai

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE -14435.804   7378.419     -1.956      0.050 -28897.506     25.897
           ATC -14330.096   7512.241     -1.908      0.056 -29054.090    393.897
           ATT -20107.151   9369.626     -2.146      0.032 -38471.618  -1742.684



Brand: Chevrolet

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE  -9441.030   4510.962     -2.093      0.036 -18282.514   -599.545
           ATC  -9442.602   4513.751     -2.092      0.036 -18289.554   -595.651
           ATT  -7271.571  13570.159     -0.536      0.592 -33869.082  19325.939



Brand: Lexus

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE -19195.736  13353.729     -1.437      0.151 -45369.044   6977.573
           ATC -22234.775  14841.213     -1.498      0.134 -51323.551   6854.002
           ATT   5736.184  16169.254      0.355      0.723 -25955.552  37427.921



Skipping Cadillac due to an error: Too few treated units: N_t < K+1
Brand: Nissan

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE  -3730.163   6087.675     -0.613      0.540 -15662.006   8201.679
           ATC  -3742.901   6094.745     -0.614      0.539 -15688.601   8202.799
           ATT   5842.125   9872.092      0.592      0.554 -13507.175  25191.425



Brand: Honda

                                            g
```

```
                      Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE  -2516.354   7365.293     -0.342      0.733 -16952.329  11919.620
           ATC  -3167.390   7554.383     -0.419      0.675 -17973.980  11639.200
           ATT  22181.276   9016.889      2.460      0.014   4508.172  39854.379
```

```
Brand: Kia

Treatment Effect Estimates: Matching

                      Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE   4145.556   5495.235      0.754      0.451  -6625.105  14916.216
           ATC   4933.803   5648.958      0.873      0.382  -6138.155  16005.761
           ATT -22703.690   8897.394     -2.552      0.011 -40142.583  -5264.797
```

```
Brand: Ford

Treatment Effect Estimates: Matching

                      Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE  19998.726   9536.223      2.097      0.036   1307.729  38689.723
           ATC  20217.763   9799.934      2.063      0.039   1009.893  39425.633
           ATT  12422.509  16245.941      0.765      0.444 -19419.536  44264.553
```

```
Brand: Lincoln

Treatment Effect Estimates: Matching

                      Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE   -251.743  10646.854     -0.024      0.981 -21119.577  20616.090
           ATC   -195.196  11041.077     -0.018      0.986 -21835.707  21445.315
           ATT  -1656.463  19645.473     -0.084      0.933 -40161.591  36848.665
```

```
Brand: Volkswagen

Treatment Effect Estimates: Matching

                      Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
           ATE -13603.810    236.280    -57.575      0.000 -14066.919 -13140.701
           ATC -13609.203    235.944    -57.680      0.000 -14071.654 -13146.753
           ATT  -9071.667  11000.173     -0.825      0.410 -30632.007  12488.673
```

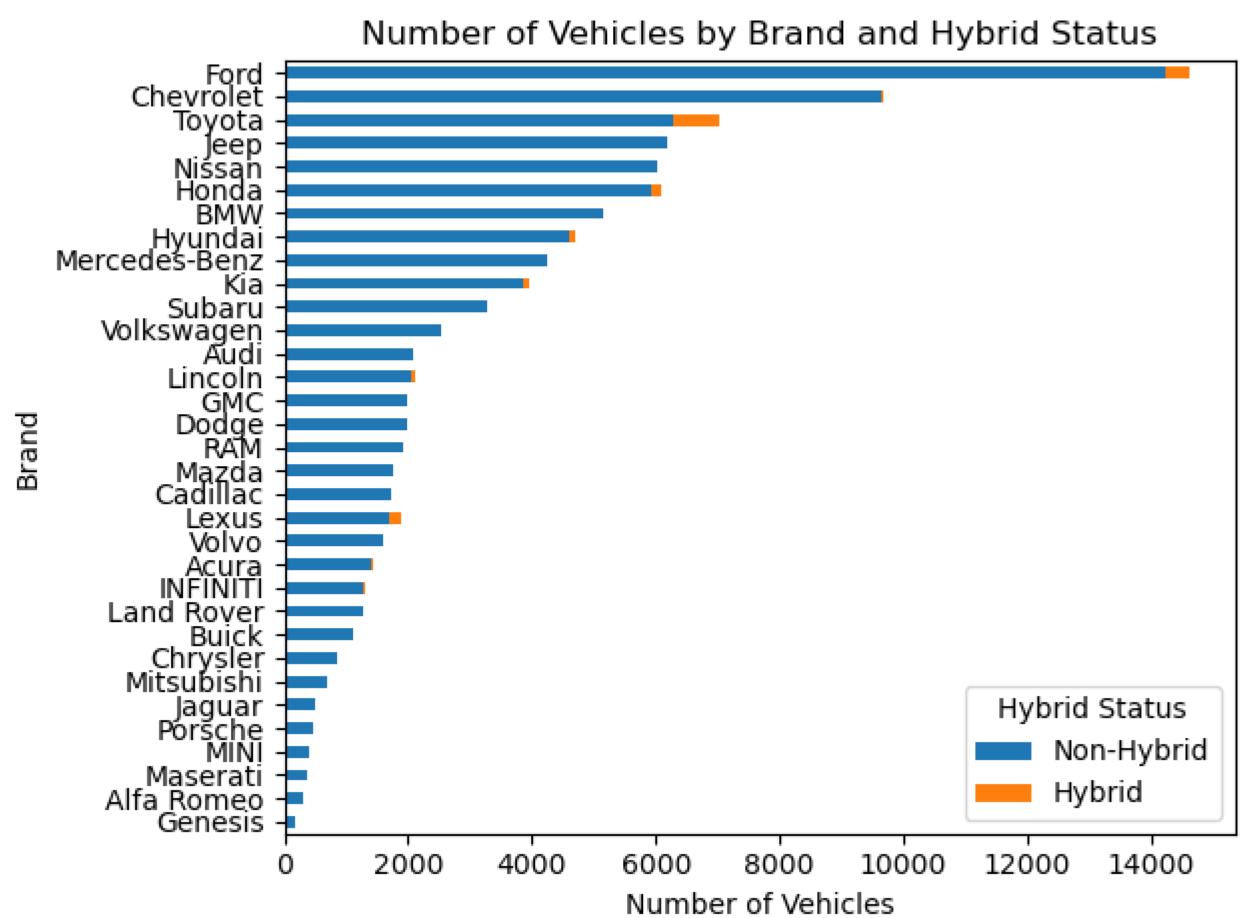
```
Brand: Porsche

Treatment Effect Estimates: Matching

                      Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
                             5    -17.004      0.000 -53515.088 -42453.176
```

```
          ATC -48117.096   2809.946    -17.124      0.000 -53624.591 -42609.601
          ATT -17402.500  60117.752     -0.289      0.772-135233.293 100428.293


Brand: Toyota

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
          ATE   9905.989  10851.811      0.913      0.361 -11363.560  31175.538
          ATC   8700.779  12053.766      0.722      0.470 -14924.602  32326.160
          ATT  20115.397  11558.417      1.740      0.082  -2539.100  42769.894


Brand: INFINITI

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
          ATE  -3482.261   5253.515     -0.663      0.507 -13779.150   6814.628
          ATC  -3516.774   5273.336     -0.667      0.505 -13852.512   6818.964
          ATT   3926.500  14628.302      0.268      0.788 -24744.971  32597.971


Brand: GMC

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
          ATE -26819.485   9250.328     -2.899      0.004 -44950.128  -8688.843
          ATC -26852.106   9254.934     -2.901      0.004 -44991.777  -8712.435
          ATT   5752.500  17816.944      0.323      0.747 -29168.709  40673.709


Brand: Acura

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
          ATE   3231.234  14553.519      0.222      0.824 -25293.663  31756.132
          ATC   3258.009  14706.083      0.222      0.825 -25565.914  32081.932
          ATT    856.625  15647.455      0.055      0.956 -29812.388  31525.638


Brand: Mercury

Treatment Effect Estimates: Matching

                     Est.       S.e.          z      P>|z|      [95% Conf. int.]
--------------------------------------------------------------------------------
          ATE  -2468.841    905.753     -2.726      0.006  -4244.117   -693.565
          ATC  -2584.619    884.186     -2.923      0.003  -4317.624   -851.614
          ATT    -37.500   4894.678     -0.008      0.994  -9631.069   9556.069
```

Looking at the scores by brand, it was found that there is a hybrid vehicle (HV) premium for the brands Acura, Toyota, Ford, and Kia. Considering the period of data collection, it cannot be asserted with statistical significance due to the small number of data on HVs. However, as a general trend, it can be considered that there is an HV premium in the US market for these four brands. This means that choosing HVs from these four brands when purchasing a new car could result in a higher resale value.

In [190…

```
# Aggregate the data based on 'make_name' and 'is_hybrid', counting the number of records for each
brand_hybrid_counts = df.groupby(['make_name', 'is_hybrid']).size().unstack(fill_value=0)

# Filter out brands with a total count of less than 100 vehicles
brand_hybrid_counts = brand_hybrid_counts[brand_hybrid_counts.sum(axis=1) >= 100]

# Sort the brands by the total number of vehicles in ascending order
# This will result in the brands with the most vehicles being at the bottom of the chart,
# which is the top when displayed as a horizontal bar chart
brand_hybrid_counts = brand_hybrid_counts.sort_values(by=[0, 1], ascending=True)

# Display the horizontal bar chart
brand_hybrid_counts.plot(kind='barh', stacked=True, color=['#1f77b4', '#ff7f0e'])

plt.title('Number of Vehicles by Brand and Hybrid Status')
plt.xlabel('Number of Vehicles')
plt.ylabel('Brand')
plt.legend(['Non-Hybrid', 'Hybrid'], title='Hybrid Status')
plt.tight_layout()

plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js