In [167…

```
# Initial imports
import pandas as pd
import hvplot.pandas
from path import Path
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import hvplot.pandas
from bokeh.sampledata.autompg import autompg_clean as df
```

In [168…

```
# Load the listings.csv dataset.
file_path = "listings.csv"
df = pd.read_csv(file_path)
df
```

Out[168…

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_r
0	2384	Hyde Park - Walk to University of Chicago	2613	Rebecca	NaN	Hyde Park	41.787900	-87.587800	Private room	85	
1	7126	Tiny Studio Apartment 94 Walk Score	17928	Sarah	NaN	West Town	41.901660	-87.680210	Entire home/apt	65	
2	10945	The Biddle House (#1)	33004	At Home Inn	NaN	Lincoln Park	41.911960	-87.639810	Entire home/apt	143	
3	12068	Chicago GOLD COAST 1 Bedroom Condo	40731	Dominic	NaN	Near North Side	41.904910	-87.632130	Entire home/apt	99	
4	12140	Lincoln Park Guest House	46734	Sharon And Robert	NaN	Lincoln Park	41.923570	-87.649470	Private room	329	

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_r
...	...	...	...	...	...	...	...	...	...	...	
6361	50917853	Stretch out. Work. Unwind. | Studio in Chicago	342643084	Kia	NaN	Near South Side	41.861608	-87.625755	Entire home/apt	131	
6362	50944574	RoomM2	327103193	Jorge	NaN	East Side	41.705821	-87.536382	Private room	22	
6363	50944692	Room M3	327103193	Jorge	NaN	East Side	41.707917	-87.538401	Private room	21	
6364	50950455	#E Comfy Private bedroom Shared bath Near Down...	401777272	Sweet Home Of Chicago	NaN	Bridgeport	41.845373	-87.646210	Private room	55	
6365	50952621	5min to Wicker, Dtown | Quiet Flat + W&D | Zen...	47172572	Zencity	NaN	West Town	41.901909	-87.690830	Entire home/apt	97	

6366 rows × 16 columns

In [169…

```
# Drop the null columns where all values are null
df = df.dropna(axis='columns', how='all')

# Drop the null rows
df = df.dropna()

# Keep the rows where price is greater than zero
df = df[df["price"]>0]

df.head()
```

Out[169…

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_revie
0	2384	Hyde Park - Walk to University of Chicago	2613	Rebecca	Hyde Park	41.78790	-87.58780	Private room	85	1	185	2021-06-

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_revie
1	7126	Tiny Studio Apartment 94 Walk Score	17928	Sarah	West Town	41.90166	-87.68021	Entire home/apt	65	2	401	2021-05-
2	10945	The Biddle House (#1)	33004	At Home Inn	Lincoln Park	41.91196	-87.63981	Entire home/apt	143	4	28	2021-06-
3	12068	Chicago GOLD COAST 1 Bedroom Condo	40731	Dominic	Near North Side	41.90491	-87.63213	Entire home/apt	99	7	11	2021-05-
4	12140	Lincoln Park Guest House	46734	Sharon And Robert	Lincoln Park	41.92357	-87.64947	Private room	329	2	7	2021-07-

In [170…

```
df = df.drop(['name', 'host_name', "last_review"], axis=1)
df.head()
```

Out[170…

	id	host_id	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_h
0	2384	2613	Hyde Park	41.78790	-87.58780	Private room	85	1	185	2.45	
1	7126	17928	West Town	41.90166	-87.68021	Entire home/apt	65	2	401	3.32	
2	10945	33004	Lincoln Park	41.91196	-87.63981	Entire home/apt	143	4	28	0.32	
3	12068	40731	Near North Side	41.90491	-87.63213	Entire home/apt	99	7	11	0.14	
4	12140	46734	Lincoln Park	41.92357	-87.64947	Private room	329	2	7	0.10	

In [171…

```
# Use get_dummies() to create variables for text features.
X = df.drop('price', axis=1)
X = pd.get_dummies(X)
#X = pd.get_dummies(df, columns=['neighbourhood','room_type'], prefix_sep='_')
X
```

Out[171…

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availa
0	2384	2613	41.78790	-87.58780	1	185	2.45	1	
1	7126	17928	41.90166	-87.68021	2	401	3.32	1	
2	10945	33004	41.91196	-87.63981	4	28	0.32	10	
3	12068	40731	41.90491	-87.63213	7	11	0.14	1	
4	12140	46734	41.92357	-87.64947	2	7	0.10	1	
…	…	…	…	…	…	…	…	…	
6277	50594581	402792711	41.69977	-87.66974	2	1	1.00	2	
6280	50613861	23612069	41.87994	-87.68348	1	1	1.00	18	
6284	50620728	1517250	41.96310	-87.68138	2	1	1.00	1	
6293	50638737	409137639	41.94426	-87.73662	1	1	1.00	1	
6300	50644921	247500723	41.81128	-87.59237	3	1	1.00	2	

5280 rows × 90 columns

In [172…

```
X.describe()
```

Out[172…

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listing
count	5.280000e+03	5.280000e+03	5280.000000	5280.000000	5280.00000	5280.000000	5280.000000	5280.
mean	2.972659e+07	1.088967e+08	41.900245	-87.665352	6.49053	52.450758	2.657919	10
std	1.437791e+07	1.119229e+08	0.060013	0.042212	20.35838	75.404381	4.773417	29
min	2.384000e+03	2.153000e+03	41.650640	-87.846720	1.00000	1.000000	0.010000	1.
25%	1.875119e+07	1.678259e+07	41.872215	-87.689098	1.00000	5.000000	0.470000	1.
50%	3.108824e+07	5.979480e+07	41.904925	-87.662600	2.00000	22.000000	1.480000	2.
75%	4.220822e+07	1.776859e+08	41.941297	-87.635795	3.00000	70.000000	3.300000	7.
max	5.064492e+07	4.099733e+08	42.022200	-87.535880	365.00000	1027.000000	121.820000	260.

8 rows × 90 columns

In [173…

```
#X.drop(columns=["name", "host_name", "last_review"], axis=1, inplace=True)
#X.head()
```

In [174…

```
# Standardize the data with StandardScaler
#X_scaled = StandardScaler().fit_transform(X)
```

In [175…

```
# Create our target
y = df['price']
```

In [176…

```
y.value_counts()
```

Out[176…

```
75      101
100      87
80       75
50       74
125      71
       ...
886       1
918       1
1046      1
1086      1
1898      1
Name: price, Length: 567, dtype: int64
```

In [181…

```
from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(X,
#    y, random_state=1, stratify=y)

X_train, X_test, y_train, y_test = train_test_split(X, y.iloc[:,1], test_size=1/3,
  random_state=85, stratify=y.iloc[:,1])
```

```
---------------------------------------------------------------------------
IndexingError                             Traceback (most recent call last)
<ipython-input-181-4a9a0a0f0eb3> in <module>
      3 #    y, random_state=1, stratify=y)
      4
----> 5 X_train, X_test, y_train, y_test = train_test_split(X, y.iloc[:,1], test_size=1/3,
      6   random_state=85, stratify=y.iloc[:,1])

~/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/pandas/core/indexing.py in __getitem__(self, key)
    887                     # AttributeError for IntervalTree get_value
    888                     return self.obj._get_value(*key, takeable=self._takeable)
--> 889             return self._getitem_tuple(key)
    890         else:
    891             # we by definition only have the 0th axis

~/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/pandas/core/indexing.py in _getitem_tuple(self, tup)
   1448     def _getitem_tuple(self, tup: Tuple):
   1449
```

```
-> 1450         self._has_valid_tuple(tup)
   1451         with suppress(IndexingError):
   1452             return self._getitem_lowerdim(tup)

~/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/pandas/core/indexing.py in _has_valid_tuple(self, key)
    718         Check the key for valid keys across my indexer.
    719         """
--> 720         self._validate_key_length(key)
    721         for i, k in enumerate(key):
    722             try:

~/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/pandas/core/indexing.py in _validate_key_length(self, key)
    759     def _validate_key_length(self, key: Sequence[Any]) -> None:
    760         if len(key) > self.ndim:
--> 761             raise IndexingError("Too many indexers")
    762
    763     def _getitem_tuple_same_dim(self, tup: Tuple):

IndexingError: Too many indexers
```

In [165…

```
classifier = LogisticRegression(solver='lbfgs',
                                max_iter=100000,
                                random_state=70,
                                stratify='y')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-165-e7838bc2f218> in <module>
      2                                 max_iter=100000,
      3                                 random_state=70,
----> 4                                 stratify='y')

~/opt/anaconda3/envs/mlenv/lib/python3.7/site-packages/sklearn/utils/validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

TypeError: __init__() got an unexpected keyword argument 'stratify'
```

In [150…

```
classifier.fit(X_train, y_train)
```

Out[150… LogisticRegression(max_iter=100000, random_state=1)

In [151…

```
y_pred = classifier.predict(X_test)
results = pd.DataFrame({"Prediction": y_pred, "Actual": y_test}).reset_index(drop=True)
results.head(20)
```

Out[151…

	Prediction	Actual
0	75	60
1	50	1095
2	75	1898
3	528	173
4	75	115
5	75	49
6	75	1125
7	75	199
8	75	90
9	75	55
10	75	249
11	75	175
12	75	52
13	75	99
14	50	125
15	75	80
16	75	148
17	75	186
18	50	786
19	50	200

In [152…

```
print(accuracy_score(y_test, y_pred))
```

```
0.015909090909090907
```