homework-01

May 19, 2024

[1]:
```
import pandas as pd
from sklearn.feature_extraction import DictVectorizer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

[2]:
```
df_jan = pd.read_parquet('https://d37ci6vzurychx.cloudfront.net/trip-data/
 ↪yellow_tripdata_2023-01.parquet')
df_feb = pd.read_parquet('https://d37ci6vzurychx.cloudfront.net/trip-data/
 ↪yellow_tripdata_2023-02.parquet')
print('Jan')
display(df_jan.head())
print('Feb')
display(df_feb.head())
```

```
Jan
```

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	\
0	2	2023-01-01 00:32:10	2023-01-01 00:40:36	1.0	
1	2	2023-01-01 00:55:08	2023-01-01 01:01:27	1.0	
2	2	2023-01-01 00:25:04	2023-01-01 00:37:49	1.0	
3	1	2023-01-01 00:03:48	2023-01-01 00:13:25	0.0	
4	2	2023-01-01 00:10:29	2023-01-01 00:21:19	1.0	

	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	\
0	0.97	1.0	N	161	141	
1	1.10	1.0	N	43	237	
2	2.51	1.0	N	48	238	
3	1.90	1.0	N	138	7	
4	1.43	1.0	N	107	79	

	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	\
0	2	9.3	1.00	0.5	0.00	0.0	
1	1	7.9	1.00	0.5	4.00	0.0	
2	1	14.9	1.00	0.5	15.00	0.0	
3	1	12.1	7.25	0.5	0.00	0.0	
4	1	11.4	1.00	0.5	3.28	0.0	

	improvement_surcharge	total_amount	congestion_surcharge	airport_fee

```
0                    1.0         14.30                   2.5         0.00
1                    1.0         16.90                   2.5         0.00
2                    1.0         34.90                   2.5         0.00
3                    1.0         20.85                   0.0         1.25
4                    1.0         19.68                   2.5         0.00
```

Feb

```
   VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0         1  2023-02-01 00:32:53   2023-02-01 00:34:34              2.0
1         2  2023-02-01 00:35:16   2023-02-01 00:35:30              1.0
2         2  2023-02-01 00:35:16   2023-02-01 00:35:30              1.0
3         1  2023-02-01 00:29:33   2023-02-01 01:01:38              0.0
4         2  2023-02-01 00:12:28   2023-02-01 00:25:46              1.0

   trip_distance  RatecodeID store_and_fwd_flag  PULocationID  DOLocationID  \
0           0.30         1.0                  N           142           163
1           0.00         1.0                  N            71            71
2           0.00         1.0                  N            71            71
3          18.80         1.0                  N           132            26
4           3.22         1.0                  N           161           145

   payment_type  fare_amount  extra  mta_tax  tip_amount  tolls_amount  \
0             2          4.4   3.50      0.5         0.0           0.0
1             4         -3.0  -1.00     -0.5         0.0           0.0
2             4          3.0   1.00      0.5         0.0           0.0
3             1         70.9   2.25      0.5         0.0           0.0
4             1         17.0   1.00      0.5         3.3           0.0

   improvement_surcharge  total_amount  congestion_surcharge  Airport_fee
0                    1.0          9.40                   2.5         0.00
1                   -1.0         -5.50                   0.0         0.00
2                    1.0          5.50                   0.0         0.00
3                    1.0         74.65                   0.0         1.25
4                    1.0         25.30                   2.5         0.00
```

Query 1: Downloading the data We'll use the same NYC taxi dataset, but instead of "Green Taxi Trip Records", we'll use "Yellow Taxi Trip Records".

Download the data for January and February 2023.

Read the data for January. How many columns are there?

[3]:
```
# Query 1
df_jan.columns.__len__()
```

[3]: 19

Query 2: Computing duration Now let's compute the duration variable. It should contain the duration of a ride in minutes.

What's the standard deviation of the trips duration in January?

[4]:
```
# Query 2
duration = pd.to_datetime(df_jan['tpep_dropoff_datetime']) - pd.
 ↪to_datetime(df_jan['tpep_pickup_datetime'])
df_jan['duration'] = duration.dt.total_seconds() / 60

duration_feb = pd.to_datetime(df_feb['tpep_dropoff_datetime']) - pd.
 ↪to_datetime(df_feb['tpep_pickup_datetime'])
df_feb['duration'] = duration_feb.dt.total_seconds() / 60
print(df_jan['duration'].std())
```

```
42.594351241920904
```

[5]:
```
df_jan.shape
```

[5]:
```
(3066766, 20)
```

Query 3: Dropping outliers Next, we need to check the distribution of the duration variable. There are some outliers. Let's remove them and keep only the records where the duration was between 1 and 60 minutes (inclusive).

What fraction of the records left after you dropped the outliers?

[6]:
```
print(f"% of data after removing outliers: {(df_jan[(df_jan['duration'] >= 1) &␣
 ↪(df_jan['duration'] <= 60)].shape[0]/df_jan.shape[0])*100}")
```

```
% of data after removing outliers: 98.1220282212598
```

[7]:
```
df_jan = df_jan[(df_jan['duration'] >= 1) & (df_jan['duration'] <= 60)].copy()
df_feb = df_feb[(df_feb['duration'] >= 1) & (df_feb['duration'] <= 60)].copy()
df_jan.shape
```

[7]:
```
(3009173, 20)
```

[8]:
```
df_jan.reset_index(drop=True, inplace=True)
df_jan.head()
```

[8]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	\
0	2	2023-01-01 00:32:10	2023-01-01 00:40:36	1.0	
1	2	2023-01-01 00:55:08	2023-01-01 01:01:27	1.0	
2	2	2023-01-01 00:25:04	2023-01-01 00:37:49	1.0	
3	1	2023-01-01 00:03:48	2023-01-01 00:13:25	0.0	
4	2	2023-01-01 00:10:29	2023-01-01 00:21:19	1.0	

	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	\
0	0.97	1.0	N	161	141	
1	1.10	1.0	N	43	237	
2	2.51	1.0	N	48	238	
3	1.90	1.0	N	138	7	
4	1.43	1.0	N	107	79	

	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	\
0	2	9.3	1.00	0.5	0.00	0.0	
1	1	7.9	1.00	0.5	4.00	0.0	
2	1	14.9	1.00	0.5	15.00	0.0	
3	1	12.1	7.25	0.5	0.00	0.0	
4	1	11.4	1.00	0.5	3.28	0.0	

	improvement_surcharge	total_amount	congestion_surcharge	airport_fee	\
0	1.0	14.30	2.5	0.00	
1	1.0	16.90	2.5	0.00	
2	1.0	34.90	2.5	0.00	
3	1.0	20.85	0.0	1.25	
4	1.0	19.68	2.5	0.00	

	duration
0	8.433333
1	6.316667
2	12.750000
3	9.616667
4	10.833333

Query 4: Let's apply one-hot encoding to the pickup and dropoff location IDs. We'll use only these two features for our model.

Turn the dataframe into a list of dictionaries (remember to re-cast the ids to strings - otherwise it will label encode them) Fit a dictionary vectorizer Get a feature matrix from it What's the dimensionality of this matrix (number of columns)?

[9]:
```
x = df_jan[["PULocationID","DOLocationID"]].copy()
x.head()
```

[9]:

	PULocationID	DOLocationID
0	161	141
1	43	237
2	48	238
3	138	7
4	107	79

[10]:
```
x.PULocationID= x.PULocationID.astype('str')
x.DOLocationID= x.DOLocationID.astype('str')
```

```
x.dtypes
```

[10]:
```
PULocationID    object
DOLocationID    object
dtype: object
```

[11]:
```
x = x.to_dict(orient='records')
```

[12]:
```
dic_vec = DictVectorizer()
X = dic_vec.fit_transform(x)
del x
X.shape
```

[12]:
```
(3009173, 515)
```

[13]:
```
print(f'No. of features: {X.shape[1]}')
```

```
No. of features: 515
```

Query 5: Training a model Now let's use the feature matrix from the previous step to train a model.

Train a plain linear regression model with default parameters Calculate the RMSE of the model on the training data What's the RMSE on train?

[14]:
```
lr = LinearRegression()
lr.fit(X, df_jan['duration'])
```

[14]:
```
LinearRegression()
```

[15]:
```
y_pred = lr.predict(X)
rmse = mean_squared_error(df_jan['duration'], y_pred, squared=False)
print(f'RMSE: {rmse}')
```

```
RMSE: 7.649261930819891
```

[16]:
```
df_feb
```

[16]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	\
0	1	2023-02-01 00:32:53	2023-02-01 00:34:34	2.0	
3	1	2023-02-01 00:29:33	2023-02-01 01:01:38	0.0	
4	2	2023-02-01 00:12:28	2023-02-01 00:25:46	1.0	
5	1	2023-02-01 00:52:40	2023-02-01 01:07:18	1.0	
6	1	2023-02-01 00:12:39	2023-02-01 00:40:36	1.0	
…	…	…	…	…	
2913950	2	2023-02-28 23:46:00	2023-03-01 00:05:00	NaN	
2913951	2	2023-02-28 23:26:02	2023-02-28 23:37:10	NaN	
2913952	2	2023-02-28 23:24:00	2023-02-28 23:38:00	NaN	

```
2913953         2  2023-02-28 23:03:00   2023-02-28 23:10:00             NaN
2913954         2  2023-02-28 23:03:03   2023-02-28 23:12:51             NaN

         trip_distance  RatecodeID store_and_fwd_flag  PULocationID  \
0                 0.30         1.0                  N           142
3                18.80         1.0                  N           132
4                 3.22         1.0                  N           161
5                 5.10         1.0                  N           148
6                 8.90         1.0                  N           137
…                  …           …                    …           …
2913950           4.65         NaN               None           249
2913951           2.47         NaN               None           186
2913952           3.49         NaN               None           158
2913953           2.13         NaN               None            79
2913954           2.28         NaN               None           161

         DOLocationID  payment_type  fare_amount  extra  mta_tax  tip_amount  \
0                 163             2         4.40   3.50      0.5        0.00
3                  26             1        70.90   2.25      0.5        0.00
4                 145             1        17.00   1.00      0.5        3.30
5                 236             1        21.90   3.50      0.5        5.35
6                 244             1        41.50   3.50      0.5        3.50
…                 …               …          …     …        …          …
2913950           140             0        20.22   0.00      0.5        4.84
2913951            79             0        13.66   0.00      0.5        2.65
2913952           143             0        17.64   0.00      0.5        0.00
2913953           162             0        13.56   0.00      0.5        2.63
2913954           140             0        14.89   0.00      0.5        3.78

         tolls_amount  improvement_surcharge  total_amount  \
0                 0.0                    1.0          9.40
3                 0.0                    1.0         74.65
4                 0.0                    1.0         25.30
5                 0.0                    1.0         32.25
6                 0.0                    1.0         50.00
…                 …                      …           …
2913950           0.0                    1.0         29.06
2913951           0.0                    1.0         20.31
2913952           0.0                    1.0         21.64
2913953           0.0                    1.0         20.19
2913954           0.0                    1.0         22.67

         congestion_surcharge  Airport_fee   duration
0                         2.5         0.00   1.683333
3                         0.0         1.25  32.083333
4                         2.5         0.00  13.300000
5                         2.5         0.00  14.633333
```

```
6                          2.5          0.00  27.950000
…                          …            …     …
2913950                    NaN           NaN  19.000000
2913951                    NaN           NaN  11.133333
2913952                    NaN           NaN  14.000000
2913953                    NaN           NaN   7.000000
2913954                    NaN           NaN   9.800000

[2855951 rows x 20 columns]
```

Query 6: Evaluating the model. RMSE on validation

[17]:
```
x_val = df_feb[["PULocationID","DOLocationID"]]
x_val.PULocationID= x_val.PULocationID.astype('str')
x_val.DOLocationID= x_val.DOLocationID.astype('str')
x_val = x_val.to_dict(orient='records')
X_val = dic_vec.transform(x_val)
y_pred = lr.predict(X_val)
rmse = mean_squared_error(df_feb['duration'], y_pred, squared=False)
print(f'Validation RMSE: {rmse}')
```

```
/tmp/ipykernel_20001/4110724250.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  x_val.PULocationID= x_val.PULocationID.astype('str')
/tmp/ipykernel_20001/4110724250.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  x_val.DOLocationID= x_val.DOLocationID.astype('str')

Validation RMSE: 7.811817675774269
```

[ ]: