Solution for Homework 1

Solution elaboared by Juan Esteban Grimaldos León

to replicate results on this workbook, set src as root

Q1

size of downloaded data First 1 Download the data:

```
In [ ]: import pandas as pd

        df = pd.read_parquet('homeworks/yellow_tripdata_2023-01.parquet', engine='pyarrow')
        print(df.shape)
```

```
(3066766, 19)
```

Q2

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

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

```
Out[ ]: Index(['VendorID', 'tpep_pickup_datetime', 'tpep_dropoff_datetime',
               'passenger_count', 'trip_distance', 'RatecodeID', 'store_and_fwd_flag',
               'PULocationID', 'DOLocationID', 'payment_type', 'fare_amount', 'extra',
               'mta_tax', 'tip_amount', 'tolls_amount', 'improvement_surcharge',
               'total_amount', 'congestion_surcharge', 'airport_fee'],
              dtype='object')
```

```
In [ ]: df['duration'] = df.tpep_dropoff_datetime - df.tpep_pickup_datetime
        df['duration'] = df['duration'].dt.total_seconds() / 60
        df['duration']
```

```
Out[ ]: 0           8.433333
        1           6.316667
        2          12.750000
        3           9.616667
        4          10.833333
                     ...
        3066761    13.983333
        3066762    19.450000
        3066763    24.516667
        3066764    13.000000
        3066765    14.400000
        Name: duration, Length: 3066766, dtype: float64
```

In [ ]:

```
df.duration.std()
```

Out[ ]:

```
42.59435124195458
```

Q3

To keep only the records where the duration was between 1 and 60 minutes (inclusive), you can use the following code:

In [ ]:

```
df_filtered = df[(df['duration'] >= 1) & (df['duration'] <= 60)].copy()
fraction =  (df_filtered.shape[0]/df.shape[0])*100
print(fraction)
```

```
98.1220282212598
```

Q4

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)?

In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['VendorID', 'tpep_pickup_datetime', 'tpep_dropoff_datetime',
       'passenger_count', 'trip_distance', 'RatecodeID', 'store_and_fwd_flag',
       'PULocationID', 'DOLocationID', 'payment_type', 'fare_amount', 'extra',
       'mta_tax', 'tip_amount', 'tolls_amount', 'improvement_surcharge',
       'total_amount', 'congestion_surcharge', 'airport_fee', 'duration'],
      dtype='object')
```

In [ ]:

```
from sklearn.feature_extraction import DictVectorizer

# Convert the dataframe into a list of dictionaries
train_dicts = df_filtered[['PULocationID', 'DOLocationID']].astype(str).to_dict(ori

# Fit a dictionary vectorizer
dv = DictVectorizer()
dv.fit(train_dicts)

# Get the feature matrix
X_train = dv.transform(train_dicts)

# Determine the number of columns in the feature matrix
```

```
num_columns = X_train.shape[1]
num_columns
```

Out[ ]:

```
515
```

In [ ]:

```
y_train = df_filtered.duration.values
```

Q5

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?

In [ ]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np


lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_train)


rmse = mean_squared_error(y_train, y_pred, squared=False)
rmse
```

```
d:\projects\juanes-grimaldos-MLops-datatalks-homework\.venv\Lib\site-packages\sklear
n\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4
and will be removed in 1.6. To calculate the root mean squared error, use the functi
on'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
7.649261931416412
```

Q6

Now let's apply this model to the validation dataset (February 2023).

What's the RMSE on validation?

In [ ]:

```
# Load the test data

categorical = ['PULocationID', 'DOLocationID']

def read_data(filename):
    df = pd.read_parquet(filename)
    df['duration'] = df.tpep_dropoff_datetime - df.tpep_pickup_datetime
    df['duration'] = df.duration.dt.total_seconds() / 60

    df = df[(df.duration >= 1) & (df.duration <= 60)].copy()

    df[categorical] = df[categorical].fillna(-1).astype('int').astype('str')

    return df
```

In [ ]:

```
df_val = read_data('homeworks/yellow_tripdata_2023-02.parquet')
```

In [ ]:

```
val_dicts = df_val[categorical].to_dict(orient='records')

X_val = dv.transform(val_dicts)

y_pred = lr.predict(X_val)

y_val = df_val.duration.values

mean_squared_error(y_val, y_pred, squared=False)
```

```
d:\projects\juanes-grimaldos-MLops-datatalks-homework\.venv\Lib\site-packages\sklear
n\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4
and will be removed in 1.6. To calculate the root mean squared error, use the functi
on'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
7.8118162035401735
```