Strategy

I chose to implement Q-learning to the problem because the size of the state space for the medium and large dataset makes it infeasible to use a model-based approach such as value or policy iteration due to the size of the transition and reward models. I implemented the Q-learning algorithm referencing Algorithm 17.2 in the textbook.
In the beginning, I initialized the values of the Q table to be 0.0, which led me to run into problems on the medium dataset. Because the medium dataset only has non-positive rewards, the model always chose an unseen action over the learned actions from the data. After changing this such that that the table is initialized with negative infinity, the model correctly selects the actions with rewards it has observed.

Performance

```
-----------------------------------------------
| dataset | execution time | raw policy score |
-----------------------------------------------
| small   | 0.912 seconds  |  27.447          |
| medium  | 1.888 seconds  |  -0.376          |
| large   | 1.836 seconds  |   0.189          |
-----------------------------------------------
```

Code

```
import sys
import pandas as pd
import numpy as np

class QLearning:
    def __init__(self, n_states, n_actions, discount, lr):
        self.n_states = n_states
        self.n_actions = n_actions
        self.discount_rate = discount # gamma
        self.Q = np.zeros((n_states, n_actions)) # action value function
        self.Q.fill(-np.inf)
        self.lr = lr # alpha: learning rate

def update(m, s, a, r, sp):
    if m.Q[s,a] == -np.inf:
        m.Q[s,a] = 0.
    m.Q[s, a] += m.lr * (r + m.discount_rate * np.max(m.Q[sp]) - m.Q[s, a])
    return m

if __name__ == '__main__':
    dataset = sys.argv[1]
```

```
    df = pd.read_csv(f'data/{dataset}.csv')
    df['s'] -= 1
    df['a'] -= 1
    df['sp'] -= 1
    if dataset == 'medium':
        model = QLearning(50000, df['a'].max()+1, 1., 0.01)
    else:
        model = QLearning(df['s'].max()+1, df['a'].max()+1, 0.95, 0.01)
    for i, obs in df.iterrows():
        model = update(model, obs['s'], obs['a'], obs['r'], obs['sp'])
    policy = np.argmax(model.Q, axis=1) + 1
    np.savetxt(f'data/{dataset}.policy', policy.astype(int), fmt='%i')
```