

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

```
In [2]: # Reading the csv
data = pd.read_csv('rawdata.csv')
```

```
In [3]: # EDA
print("##### Shape #####")
print(data.shape)

print("##### Types #####")
print(data.dtypes)
```

```
##### Shape #####
(1202783, 11)
##### Types #####
patient_token    object
supply_date      object
supply_time      object
pharmacy_id      object
birth_year       int64
gender           int64
cnk              int64
product_name     object
atc              object
units            int64
price            float64
dtype: object
```

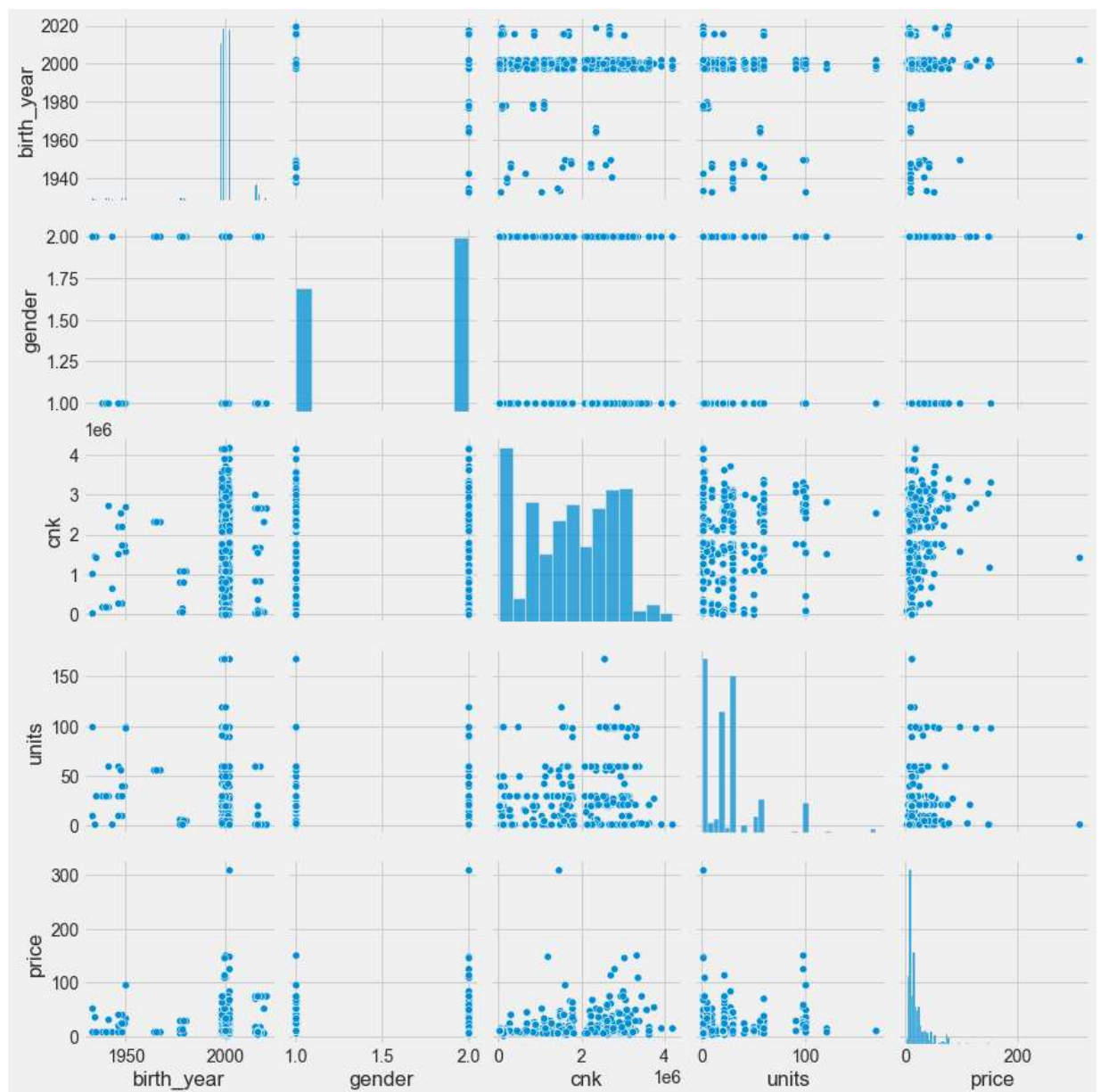
```
In [4]: data.head()
```

```
Out[4]:
```

| | patient_token | supply_date | supply_time | pharmacy_id | birth_year | gender | cnk | product_name |
|---|---------------|-------------|-------------|-------------|------------|--------|---------|-----------------------------------|
| 0 | a | 02/01/2017 | 17:14 | A2 | 2016 | 2 | 62521 | NYSTAT SUSP 1X2, 100000U |
| 1 | a | 02/01/2017 | 17:14 | A2 | 2016 | 2 | 62521 | NYSTAT SUSP 1X2, 100000U |
| 2 | a | 08/01/2020 | 15:13 | A3 | 2019 | 1 | 62521 | NYSTAT SUSP 1X2, 100000U |
| 3 | a | 09/01/2017 | NaN | A1 | 2016 | 2 | 1543305 | ATROV MONOD 0,25MG/ VIAL |
| 4 | a | 23/01/2020 | 11:21 | A2 | 2019 | 1 | 2322436 | ROTATE TUBE 1 D = 2 |

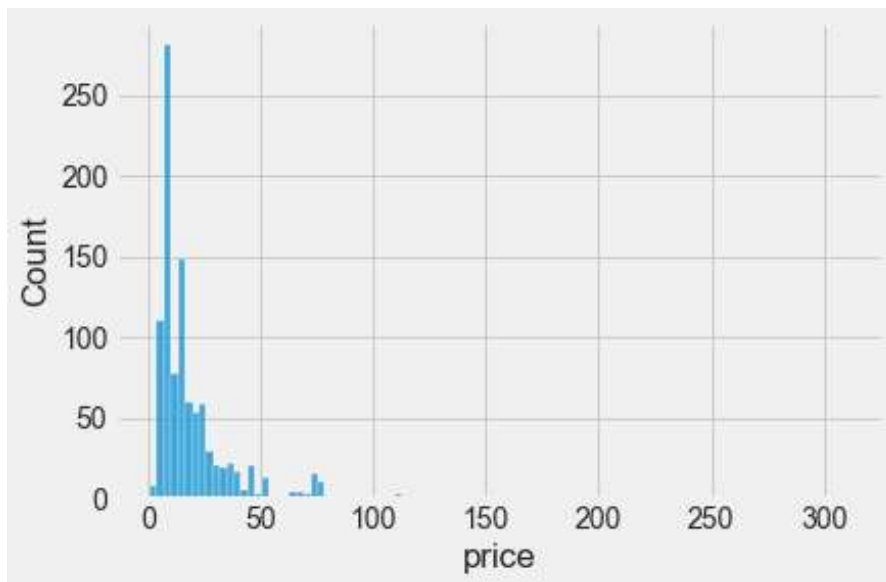
```
In [5]: data = data.head(1000)
sns.pairplot(data)
```

```
Out[5]: <seaborn.axisgrid.PairGrid at 0x7ff7d800acd0>
```



```
In [33]: sns.histplot(data['price'])
```

```
Out[33]: <AxesSubplot:xlabel='price', ylabel='Count'>
```



```
In [10]: sns.heatmap(data.corr(), annot=True)
```

```
Out[10]: <AxesSubplot:>
```



```
In [12]: # Training a Linear Regression Model
X = data[['birth_year', 'cnk', 'gender', 'units']]
y = data['price']
```

```
In [13]: # Train Test Split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
```

```
In [14]: from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, X, y, cv=10)
    return pred.mean()

def print_evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    print('MAE:', mae)
```

```

print('MSE:', mse)
print('RMSE:', rmse)
print('R2 Square', r2_square)

def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square

```

```

In [17]: # Preparing Data for Linear regression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

pipeline = Pipeline([('std_scaler', StandardScaler())])

X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)

```

```

In [18]: # Linear regression

from sklearn.linear_model import LinearRegression

lreg = LinearRegression(normalize=True)
lreg.fit(X_train, y_train)

```

Out[18]: LinearRegression(normalize=True)

```

In [19]: # Model Evaluation
print(lreg.intercept_)

```

20.106

```

In [20]: coeff_df = pd.DataFrame(lreg.coef_, X.columns, columns=['Coefficient'])
coeff_df

```

```

Out[20]:
      Coefficient
birth_year    1.200177
cnk           5.654222
gender       -2.570862
units        -0.739471

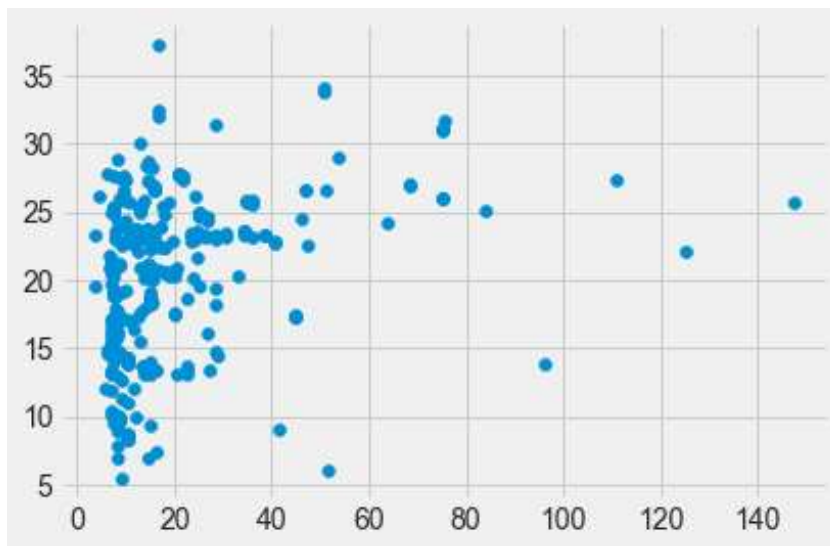
```

```

In [23]: pred = lreg.predict(X_test)
plt.scatter(y_test, pred)

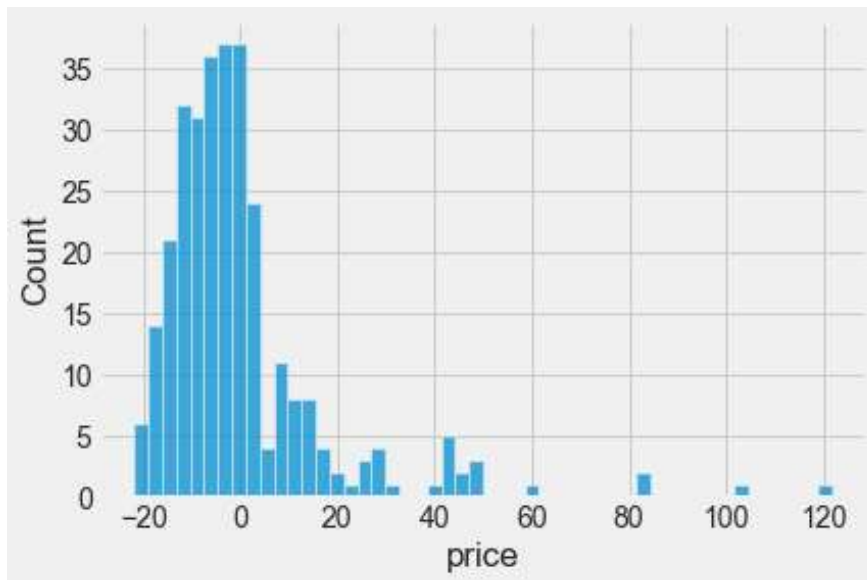
```

Out[23]: <matplotlib.collections.PathCollection at 0x7ff81e0f1730>



```
In [34]: sns.histplot((y_test - pred), bins=50)
```

```
Out[34]: <AxesSubplot:xlabel='price', ylabel='Count'>
```



```
In [37]: test_pred = lreg.predict(X_test)
train_pred = lreg.predict(X_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test, test_pred)
print('=====')
print('Train set evaluation:\n_____')
print_evaluate(y_train, train_pred)
```

Test set evaluation:

MAE: 11.183837694513857
MSE: 323.47624776768066
RMSE: 17.98544544257052
R2 Square 0.1082905247253324

=====

Train set evaluation:

MAE: 11.852840092064808
MSE: 415.0237898989758
RMSE: 20.372132679201158
R2 Square 0.07811602251535754

```
In [39]: # MAE is the easiest to understand, because it's the average error.
```

```

# MSE is more popular than MAE, because MSE "punishes" larger errors, which tends
# RMSE is even more popular than MSE, because RMSE is interpretable in the "y" uni

results_df = pd.DataFrame(data=["Linear Regression", *evaluate(y_test, test_pred)
                               columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cr
results_df

```

Out[39]:

| | Model | MAE | MSE | RMSE | R2 Square | Cross Validation |
|---|-------------------|-----------|------------|-----------|-----------|------------------|
| 0 | Linear Regression | 11.183838 | 323.476248 | 17.985445 | 0.108291 | -0.399531 |