- دریافت فایل داده ها (data set)

یک فایل csv که دارای ۱۵ ستون میباشد که به ترتیب از چپ به راست اطلاعات زیر را شامل میشوند:

- ✓ تاریخ
- ✓ نام نماد
- ✓ نوع امنیت
- ✓ قیمت لحظه بسته شدن روز قبل
- ✓ قیمت بازگشایی روز
- ✓ بیشترین قیمت روز
- ✓ کمترین قیمت روز
- ✓ قیمت آخرین معامله
- ✓ قیمت لحظه بسته شدن
- ✓ حجم وزنی میانگین قیمت
- ✓ حجم
- ✓ حجم معاملات
- ✓ معاملات (خالی میباشد)
- ✓ حجم تحویل
- ✓ درصد تحویل

- ذخیره کردن اطلاعات هر ستون در یک لیست و نمایش نمودارهای قیمت بازگشایی ، قیمت آخرین معامله و قیمت هنگام بسته شدن (open, last, close) برای شفاف سازی (visualization) داده ها.

```
#----------------------read data from file--------------------------
data = pandas.read_csv('ADANIPORTS.csv')

date, lastClose, open, high, low, last, close, volume = data['Date'], data['Prev Close'], data['Open'], data['High'], data['Low'], data['Last'], data['Close'], data['Volume']

#----------------------normalize values--------------------------
lastClose = normalize(lastClose)
open = normalize(open)
high = normalize(high)
low = normalize(low)
last = normalize(last)
close = normalize(close)
volume = normalize(volume)

#----------------------visualization--------------------------
plt.subplot(1,3,1)
plt.plot(open)
plt.ylabel('price')
plt.title('open')

plt.subplot(1,3,2)
plt.plot(last)
plt.title('last')

plt.subplot(1,3,3)
plt.plot(close)
plt.title('close')

plt.show()
```

- نرمال سازی داده ها و ساخت یک لیست برای نگه داری feature ها و label ها به نام features که در هر ایندکس آن به ترتیب قیمت بسته شدن روز قبل ، بالا ترین قیمت ، پایین ترین قیمت روز ، قیمت آخرین معامله ، قیمت لحظه بسته شدن میباشد. (با توجه به نتیجه پیش بینی عدم استفاده از بالاترین و پایین ترین قیمت باعث نتیجه بهتر میشود.)

```
#----------------------create and set value for features and labels--------------------------

features = []
label = []
for i in range(len(date)): # if we dont add high and low to featurs, give better result
    tempList = []
    tempList.append(lastClose[i])
    # tempList.append(high[i])
    # tempList.append(low[i])
    tempList.append(last[i])
    tempList.append(close[i])
    if close[i] - lastClose[i] >= 0:
        label.append('Positive')
    else:
        label.append('Negative')
    features.append(tempList)



features = np.array(features)
y = np.array(label)
```

- تقسیم کردن داده ها به دو دسته train و test که 80 درصد داده ها برای train و باقی برای test استفاده میشوند.

```
#----------------------create test array and train array (80% of data => train)--------------------------

X_train, X_test, y_train, y_test = train_test_split(features, y, test_size=0.2)
print('shape of y_train:', y_train.shape)
print('shape of y_train:', y_test.shape)
```

متد train_test_split عمل جداسازی داده ها را انجام میدهد که نتیجه را در آرایه های متناظر ذخیره میکند ، که در این متد به طور پیش فرض داده ها را پخش و جا به جا (shuffle) میکند.

- استفاده از الگوریتم KNN (k-nearest neighbors) برای پیش بینی که به ازای مقدار k از ۱ تا ۳۲ مقادیر مختلف در یک دیکشنری ذخیره شدند.

به ازای هر k مقدار accuracy توسط متد accuracy_score بدست آمده و چاپ شده اند.

```
#-----------------------use KNN algorithm and predict values--------------------------
scores = {}

for i in range(1,31):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred = knn.predict(X_test)

    scores[i] = metrics.accuracy_score(y_test, pred)

print('result: (k : predicted result)')
print(scores)
```