outliers

September 29, 2021

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
sns.set()
```

[2]:
```
tesla = pd.read_csv('../dataset/TSLA.csv')
tesla['Date'] = pd.to_datetime(tesla['Date'])
tesla.head()
```

[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-03-23	311.250000	311.250000	300.450012	301.540009	301.540009	6654900
1	2018-03-26	307.339996	307.589996	291.359985	304.179993	304.179993	8375200
2	2018-03-27	304.000000	304.269989	277.179993	279.179993	279.179993	13872000
3	2018-03-28	264.579987	268.679993	252.100006	257.779999	257.779999	21001400
4	2018-03-29	256.489990	270.959991	248.210007	266.130005	266.130005	15170700

[3]:
```
def df_shift(df,lag=0, start=1, skip=1, rejected_columns = []):
    df = df.copy()
    if not lag:
        return df
    cols ={}
    for i in range(start,lag+1,skip):
        for x in list(df.columns):
            if x not in rejected_columns:
                if not x in cols:
                    cols[x] = ['{}_{}'.format(x, i)]
                else:
                    cols[x].append('{}_{}'.format(x, i))
    for k,v in cols.items():
```

```
        columns = v
        dfn = pd.DataFrame(data=None, columns=columns, index=df.index)
        i = (skip - 1)
        for c in columns:
            dfn[c] = df[k].shift(periods=i)
            i+=skip
        df = pd.concat([df, dfn], axis=1, join_axes=[df.index])
    return df
```

[4]:
```
tesla = tesla[['Date','Close']]
tesla.head(1)
```

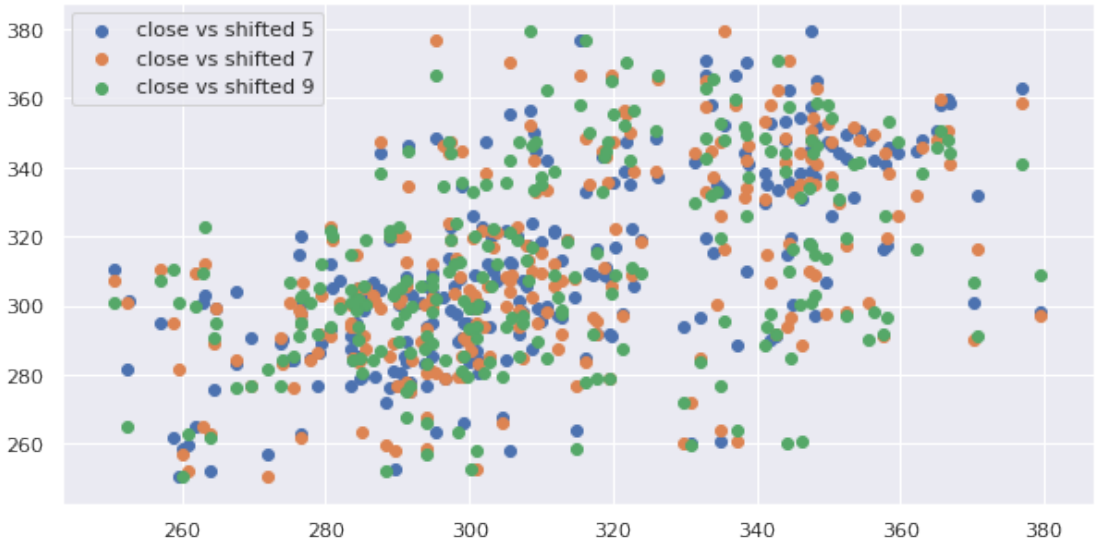
[4]:
```
         Date       Close
0 2018-03-23  301.540009
```

[5]:
```
df_crosscorrelated = df_shift(tesla, lag = 10, start = 1, skip =␣
↪2,rejected_columns=['Date'])
df_crosscorrelated['ma7'] = df_crosscorrelated['Close'].rolling(7).mean()
df_crosscorrelated['ma14'] = df_crosscorrelated['Close'].rolling(14).mean()
df_crosscorrelated['ma25'] = df_crosscorrelated['Close'].rolling(25).mean()
df_crosscorrelated.head(10)
```

[5]:
```
         Date       Close     Close_1     Close_3     Close_5     Close_7  \
0 2018-03-23  301.540009         NaN         NaN         NaN         NaN
1 2018-03-26  304.179993  301.540009         NaN         NaN         NaN
2 2018-03-27  279.179993  304.179993         NaN         NaN         NaN
3 2018-03-28  257.779999  279.179993  301.540009         NaN         NaN
4 2018-03-29  266.130005  257.779999  304.179993         NaN         NaN
5 2018-04-02  252.479996  266.130005  279.179993  301.540009         NaN
6 2018-04-03  267.529999  252.479996  257.779999  304.179993         NaN
7 2018-04-04  286.940002  267.529999  266.130005  279.179993  301.540009
8 2018-04-05  305.720001  286.940002  252.479996  257.779999  304.179993
9 2018-04-06  299.299988  305.720001  267.529999  266.130005  279.179993

      Close_9         ma7  ma14  ma25
0         NaN         NaN   NaN   NaN
1         NaN         NaN   NaN   NaN
2         NaN         NaN   NaN   NaN
3         NaN         NaN   NaN   NaN
4         NaN         NaN   NaN   NaN
5         NaN         NaN   NaN   NaN
6         NaN  275.545713   NaN   NaN
7         NaN  273.459998   NaN   NaN
8         NaN  273.679999   NaN   NaN
9  301.540009  276.554284   NaN   NaN
```

[6]:
```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
plt.scatter(df_crosscorrelated['Close'],df_crosscorrelated['Close_5'])
plt.title('close vs shifted 5')
plt.subplot(1,3,2)
plt.scatter(df_crosscorrelated['Close'],df_crosscorrelated['Close_7'])
plt.title('close vs shifted 7')
plt.subplot(1,3,3)
plt.scatter(df_crosscorrelated['Close'],df_crosscorrelated['Close_9'])
plt.title('close vs shifted 9')
plt.show()
```

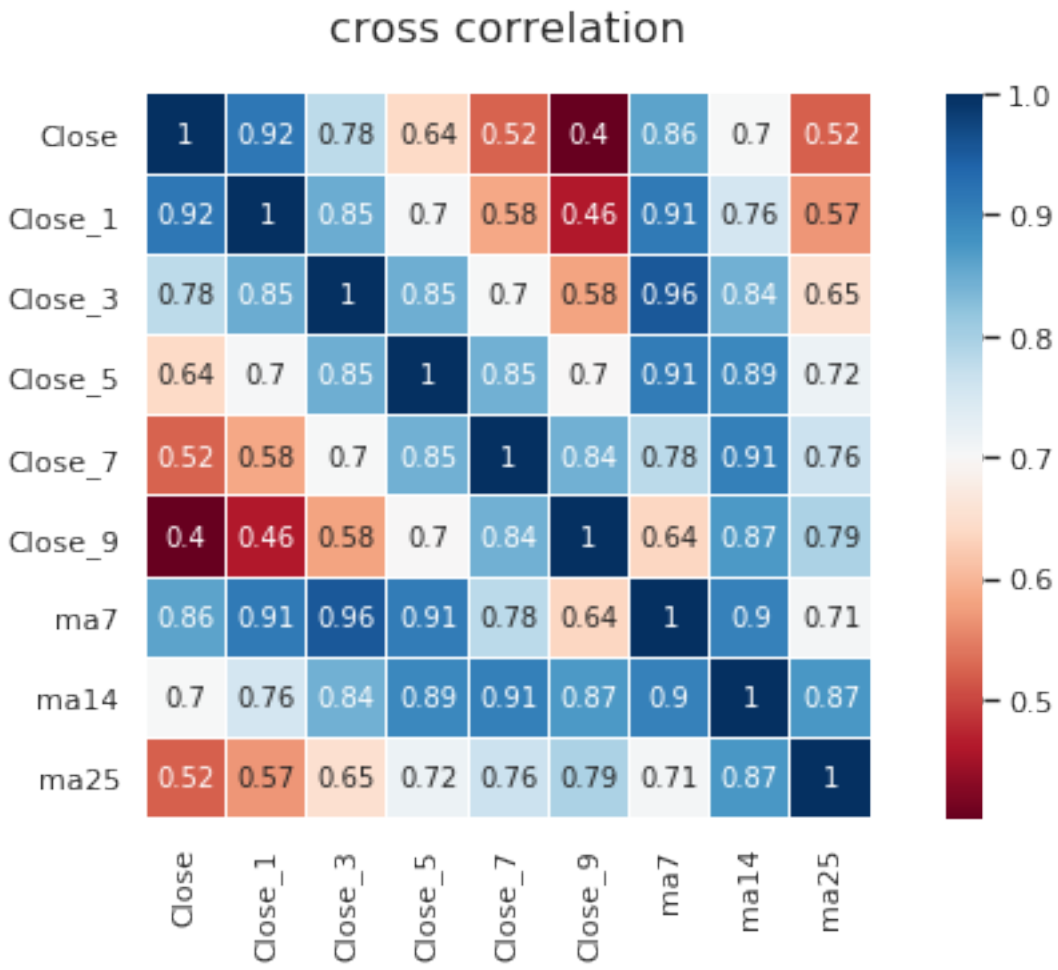
[7]:
```
plt.figure(figsize=(10,5))
plt.
 ↪scatter(df_crosscorrelated['Close'],df_crosscorrelated['Close_5'],label='close␣
 ↪vs shifted 5')
plt.
 ↪scatter(df_crosscorrelated['Close'],df_crosscorrelated['Close_7'],label='close␣
 ↪vs shifted 7')
plt.
 ↪scatter(df_crosscorrelated['Close'],df_crosscorrelated['Close_9'],label='close␣
 ↪vs shifted 9')
plt.legend()
plt.show()
```



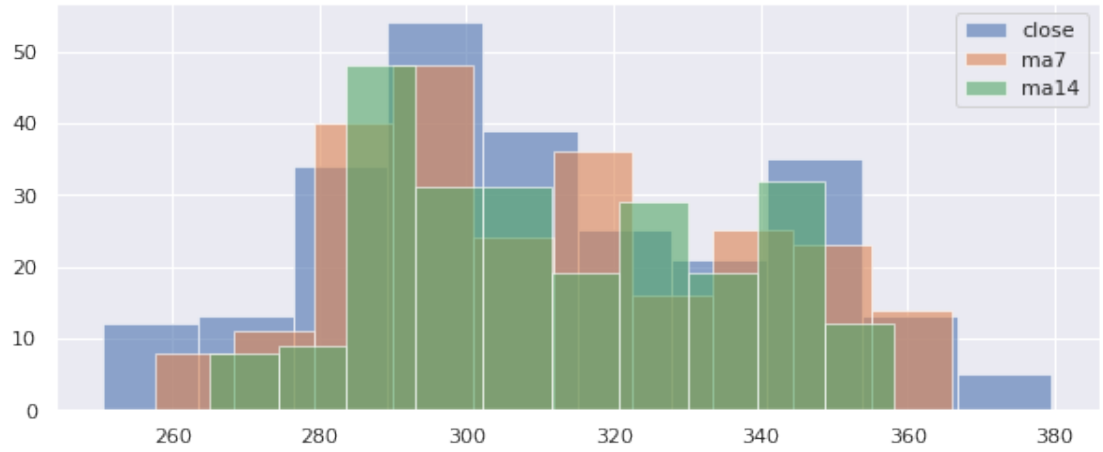
[8]:
```
fig, ax = plt.subplots(figsize=(10,4))
df_crosscorrelated.plot(x='Date',y=['Close','ma7','ma14','ma25'],ax=ax)
plt.show()
```

[9]:
```
colormap = plt.cm.RdBu
plt.figure(figsize=(10, 5))
ax=plt.subplot(111)
plt.title('cross correlation', y=1.05, size=16)
selected_column =␣
↪['Close','Close_1','Close_3','Close_5','Close_7','Close_9','ma7','ma14','ma25']
```

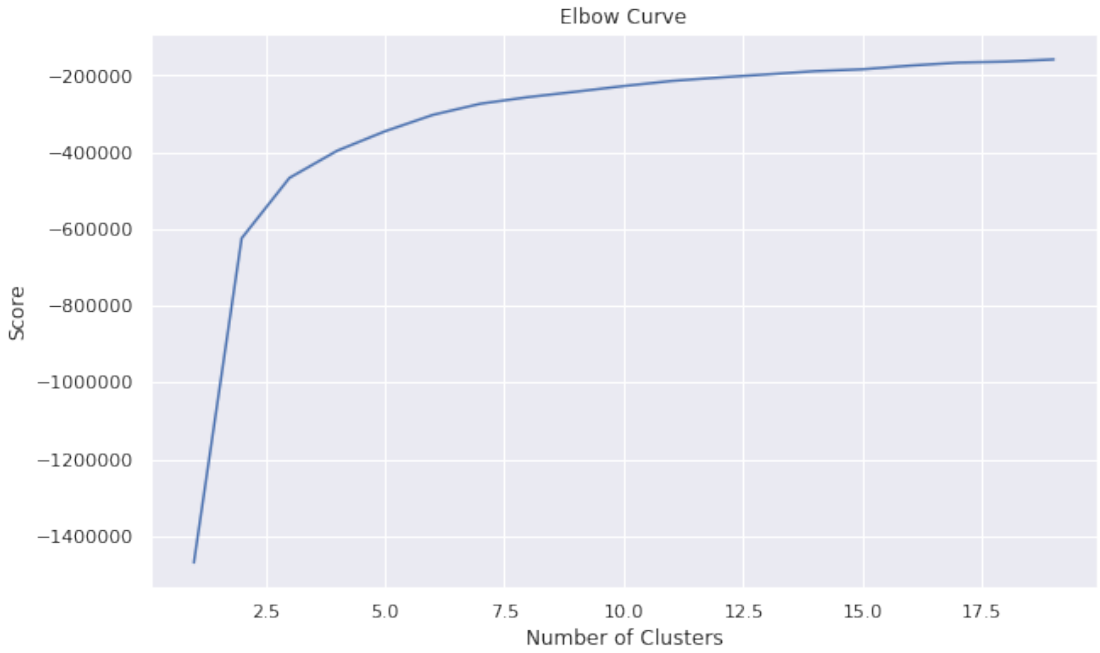
```
sns.heatmap(df_crosscorrelated[selected_column].corr(), ax=ax, linewidths=0.
↪1,vmax=1.0,
            square=True, cmap=colormap, linecolor='white', annot=True)
plt.show()
```



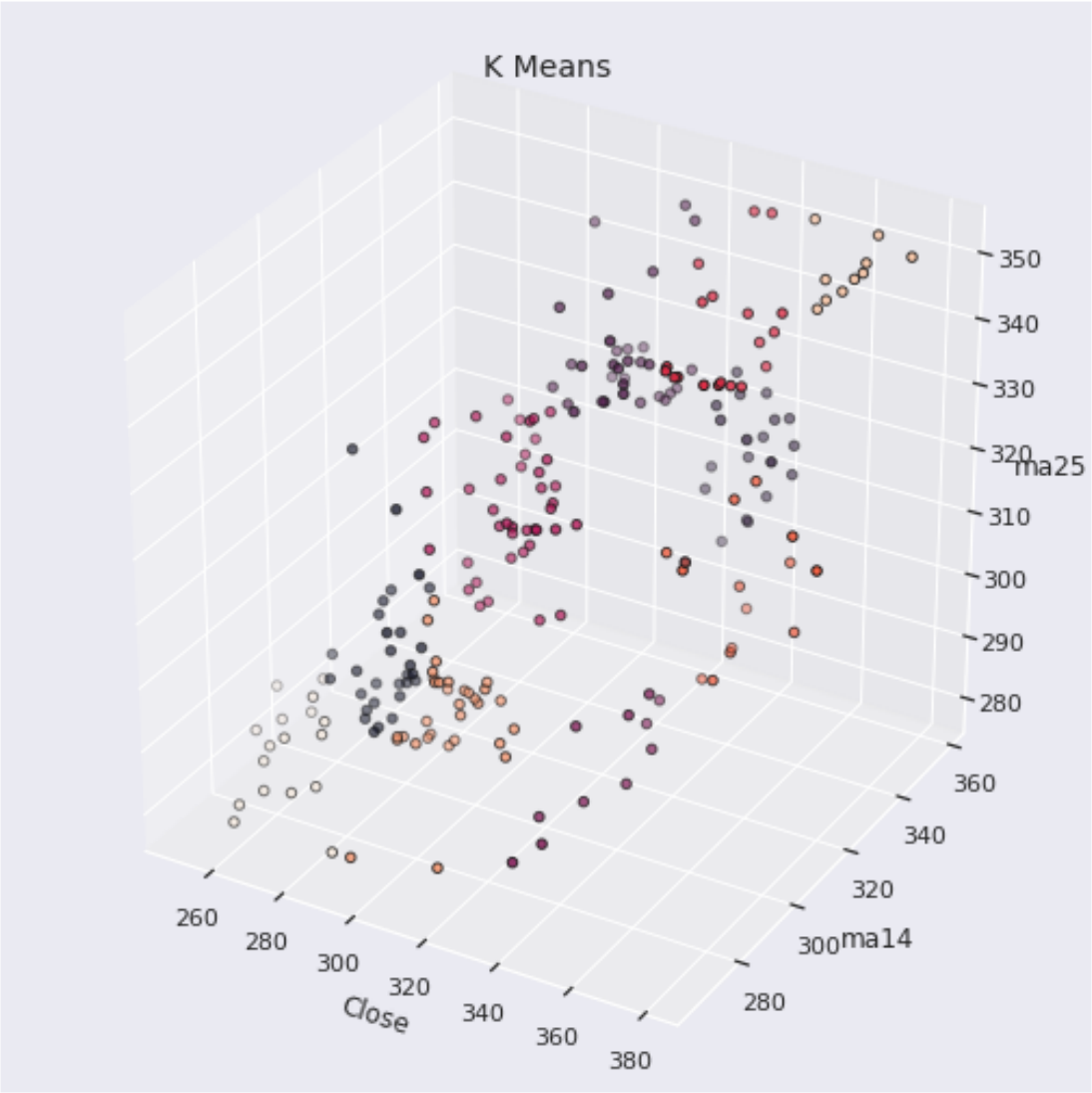
[10]:
```
fig, ax = plt.subplots(figsize=(10,4))
df_crosscorrelated['Close'].hist(alpha=0.6,label='close',ax=ax)
df_crosscorrelated['ma7'].hist(alpha=0.6,label='ma7',ax=ax)
df_crosscorrelated['ma14'].hist(alpha=0.6,label='ma14',ax=ax)
plt.legend()
plt.show()
```



[11]:
```
from sklearn.cluster import KMeans

n_cluster = range(1, 20)
data = df_crosscorrelated.iloc[:,1:].dropna().values
kmeans = [KMeans(n_clusters=i).fit(data) for i in n_cluster]
scores = [kmeans[i].score(data) for i in range(len(kmeans))]

fig, ax = plt.subplots(figsize=(10,6))
ax.plot(n_cluster, scores)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

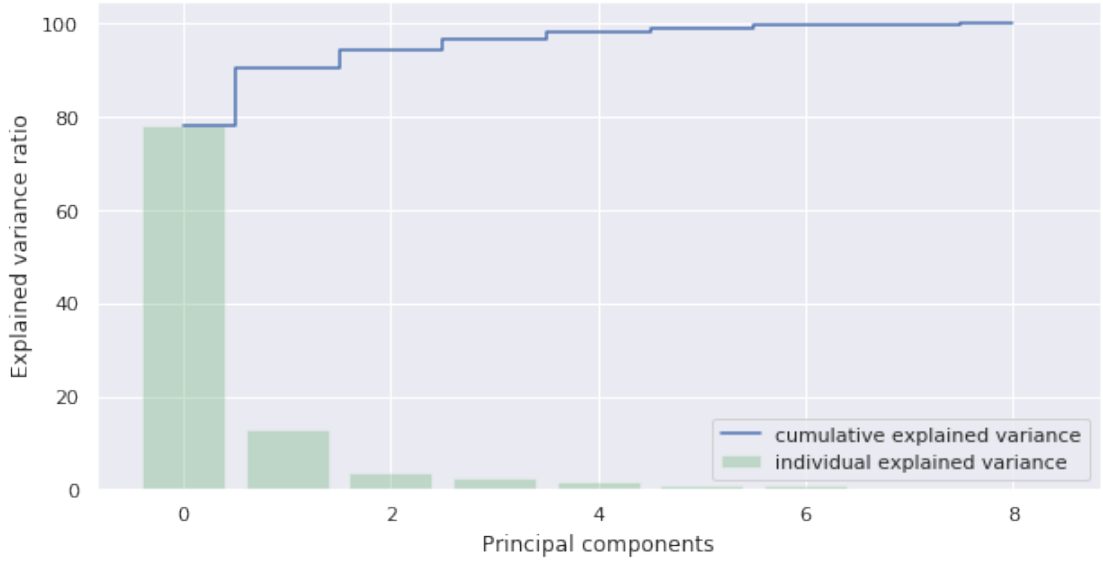


[12]:
```
from mpl_toolkits.mplot3d import Axes3D

X = df_crosscorrelated[['Close','ma14','ma25']].dropna()
X = X.reset_index(drop=True)
km = KMeans(n_clusters=10)
km.fit(X)
km.predict(X)
labels = km.labels_

fig = plt.figure(1, figsize=(7,7))
ax = Axes3D(fig)
ax.scatter(X.iloc[:,0], X.iloc[:,1], X.iloc[:,2],
           c=labels.astype(np.float), edgecolor="k")
ax.set_xlabel("Close")
ax.set_ylabel("ma14")
ax.set_zlabel("ma25")
plt.title("K Means", fontsize=14)
plt.show()
```



[13]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

X = df_crosscorrelated.iloc[:,1:].dropna().values
X_std = StandardScaler().fit_transform(X)

mean_vec = np.mean(X_std, axis=0)
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)

eig_pairs = [(np.abs(eig_vals[i]),eig_vecs[:,i]) for i in range(len(eig_vals))]

eig_pairs.sort(key = lambda x: x[0], reverse= True)
```

```
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)

plt.figure(figsize=(10, 5))
plt.bar(range(len(var_exp)), var_exp, alpha=0.3, align='center',␣
 ↪label='individual explained variance', color = 'g')
plt.step(range(len(cum_var_exp)), cum_var_exp, where='mid',label='cumulative␣
 ↪explained variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.show();
```



[14]:
```
X = df_crosscorrelated.iloc[:,1:].dropna().values
X_std = StandardScaler().fit_transform(X)
data = pd.DataFrame(X_std)
pca = PCA(n_components=2)
data = pca.fit_transform(data)
scaler = StandardScaler()
np_scaled = scaler.fit_transform(data)
```

[15]:
```
df = df_crosscorrelated.dropna()
kmeans = KMeans(n_clusters=10).fit(np_scaled)
df['cluster'] = kmeans.predict(np_scaled)
df = df.reset_index()
```

```
df['principal_feature1'] = np_scaled[:,0]
df['principal_feature2'] = np_scaled[:,1]
df['cluster'].value_counts()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.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: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

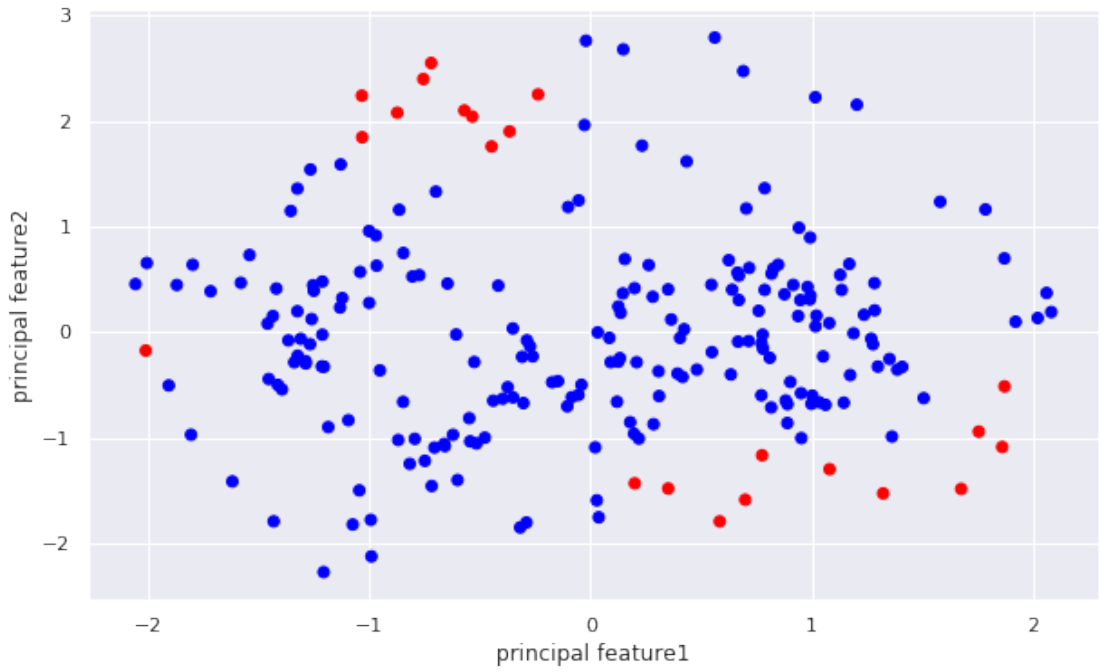
[15]:
```
8    38
7    35
3    35
4    29
5    24
2    21
6    13
1    13
0    12
9     7
Name: cluster, dtype: int64
```

[16]:
```
def getDistanceByPoint(data, model):
    distance = pd.Series()
    for i in range(0,len(data)):
        Xa = data[i]
        Xb = model.cluster_centers_[model.labels_[i]-1]
        distance.set_value(i, np.linalg.norm(Xa-Xb))
    return distance

outliers_fraction = 0.1
distance = getDistanceByPoint(np_scaled, kmeans)
number_of_outliers = int(outliers_fraction*len(distance))
threshold = distance.nlargest(number_of_outliers).min()
df['anomaly1'] = (distance >= threshold).astype(int)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: FutureWarning:
set_value is deprecated and will be removed in a future release. Please use
.at[] or .iat[] accessors instead
```

[17]:
```
fig, ax = plt.subplots(figsize=(10,6))
colors = {0:'blue', 1:'red'}
```

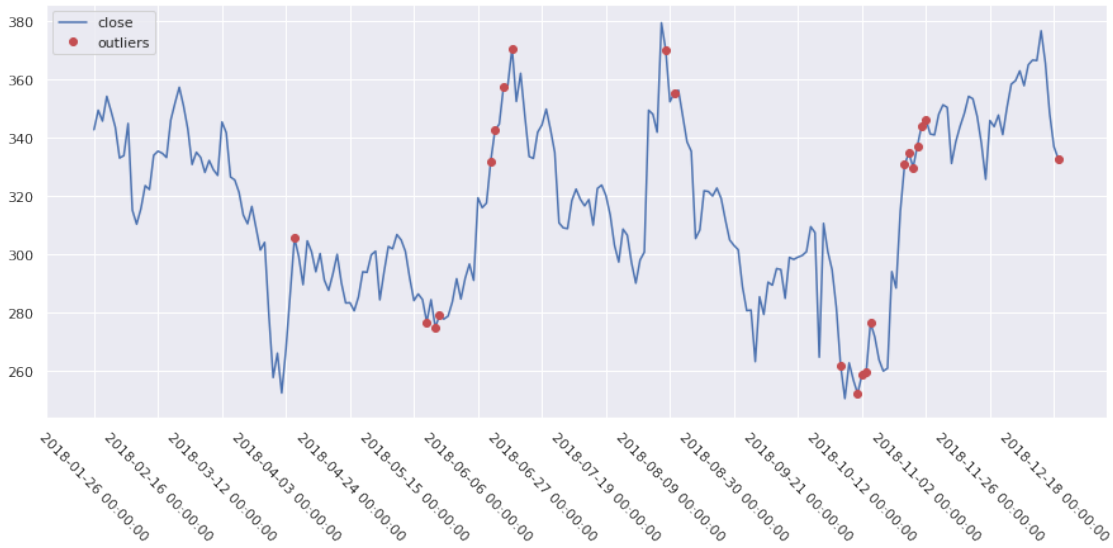
```
ax.scatter(df['principal_feature1'], df['principal_feature2'], c=df["anomaly1"].
 ↪apply(lambda x: colors[x]))
plt.xlabel('principal feature1')
plt.ylabel('principal feature2')
plt.show()
```



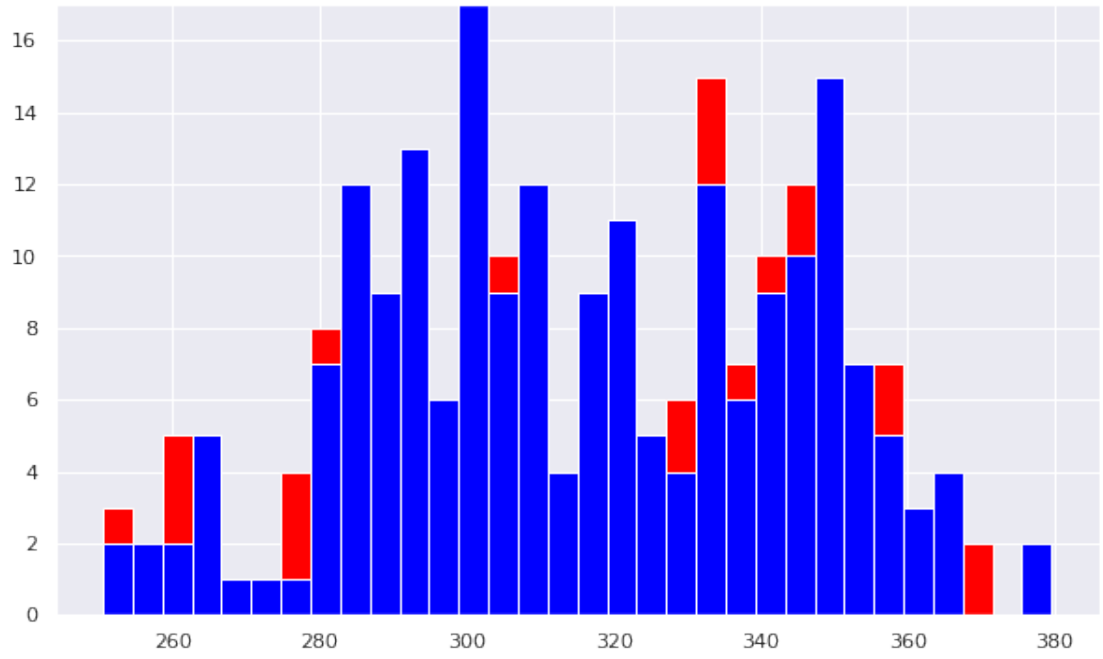
[18]:
```
df.anomaly1.value_counts()
```

[18]:
```
0    205
1     22
Name: anomaly1, dtype: int64
```

[51]:
```
plt.figure(figsize=(15, 6))
plt.plot(df['Close'], label='close',c='b')
plt.plot(df['Close'], 'o', label='outliers',markevery=df.loc[df['anomaly1'] ==␣
 ↪1].index.tolist(),c='r')
plt.xticks(np.arange(df.shape[0])[::15],df['Date'][::15],rotation='-45')
plt.legend()
plt.show()
```



[42]:
```
a = df.loc[df['anomaly1'] == 0, 'Close']
b = df.loc[df['anomaly1'] == 1, 'Close']

fig, axs = plt.subplots(figsize=(10,6))
axs.hist([a,b], bins=32, stacked=True, color=['blue', 'red'])
plt.show()
```



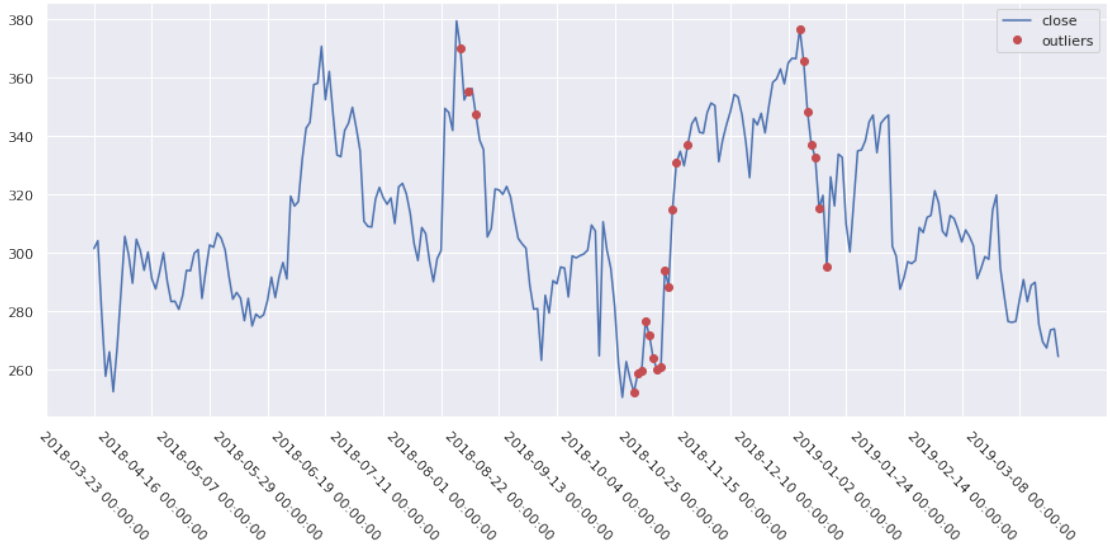
[33]:
```
ori_len = df_crosscorrelated.shape[0] - X.shape[0]
ori_len
```

[33]: 24

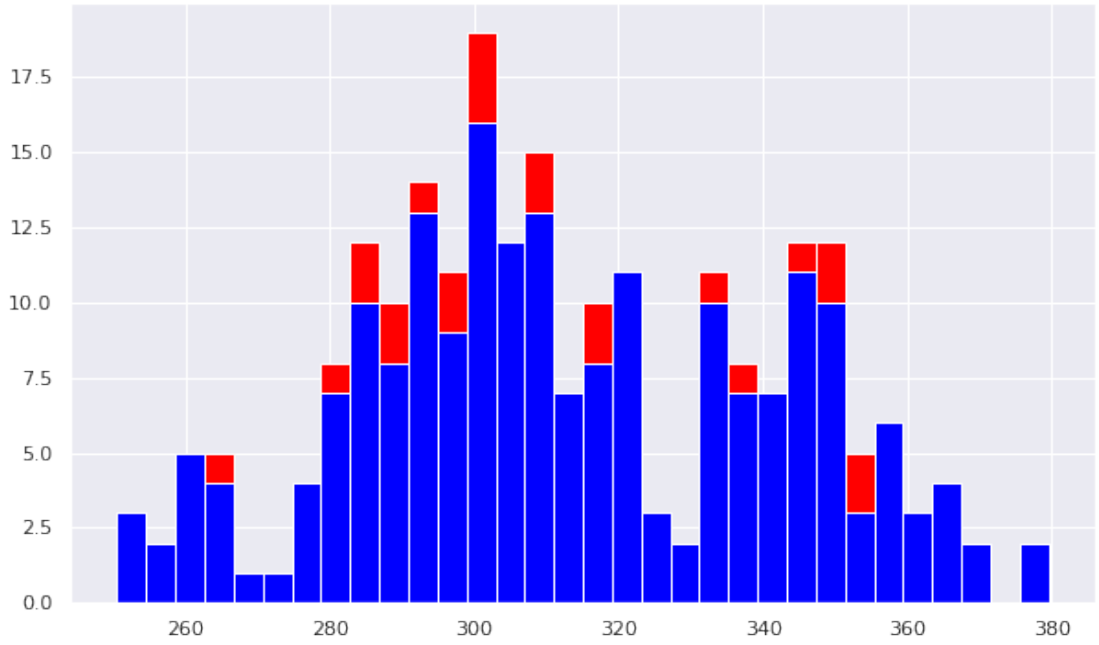
[30]:
```
np.where(outliers==-1)[0] + ori_len
```

[30]:
```
array([ 95,  97, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150,
       151, 152, 154, 156, 183, 184, 185, 186, 187, 190])
```

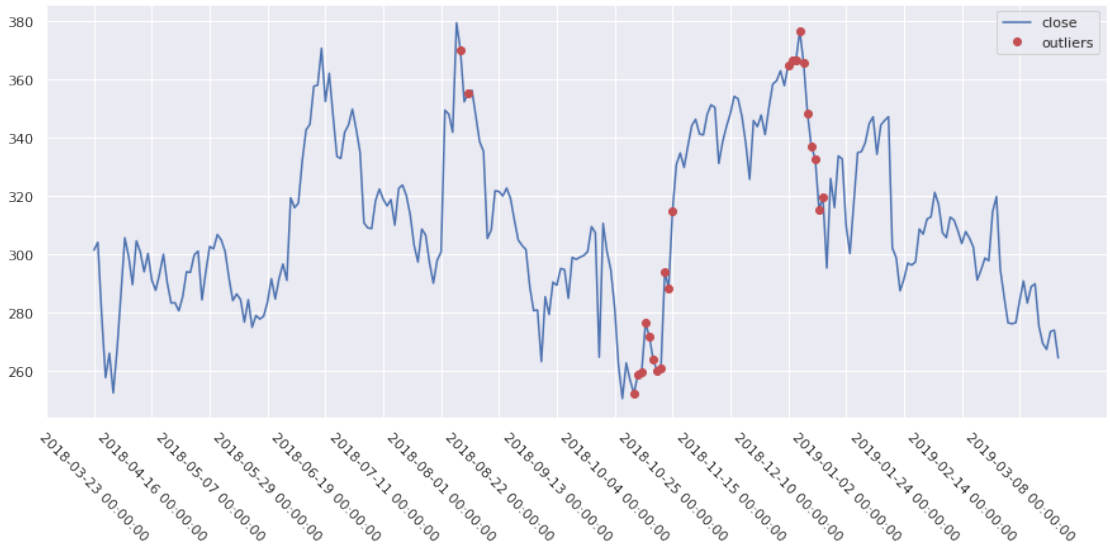
[34]:
```
from sklearn.ensemble import IsolationForest

X = df_crosscorrelated.iloc[:,1:].dropna().values
np_scaled = StandardScaler().fit_transform(X)

model = IsolationForest(contamination=outliers_fraction)
model.fit(np_scaled)
outliers = model.predict(np_scaled)

plt.figure(figsize=(15, 6))
plt.plot(df_crosscorrelated['Close'], label='close',c='b')
plt.plot(df_crosscorrelated['Close'], 'o', label='outliers',
         markevery=(np.where(outliers==-1)[0] + ori_len).tolist(),c='r')
plt.xticks(np.arange(df_crosscorrelated.shape[0])[::
 ↪15],df_crosscorrelated['Date'][::15],rotation='-45')
plt.legend()
plt.show()
```



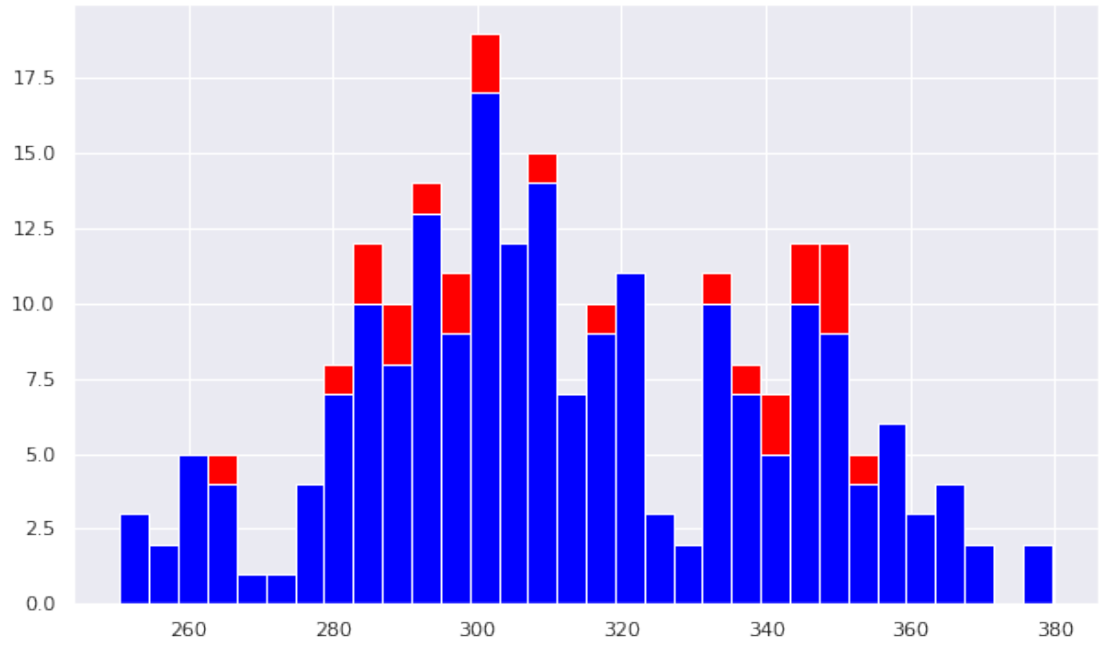
[35]:
```
close = df_crosscorrelated['Close'].values
a = close[np.where(outliers==1)[0]]
b = close[np.where(outliers==-1)[0]]

fig, axs = plt.subplots(figsize=(10,6))
axs.hist([a,b], bins=32, stacked=True, color=['blue', 'red'])
plt.show()
```



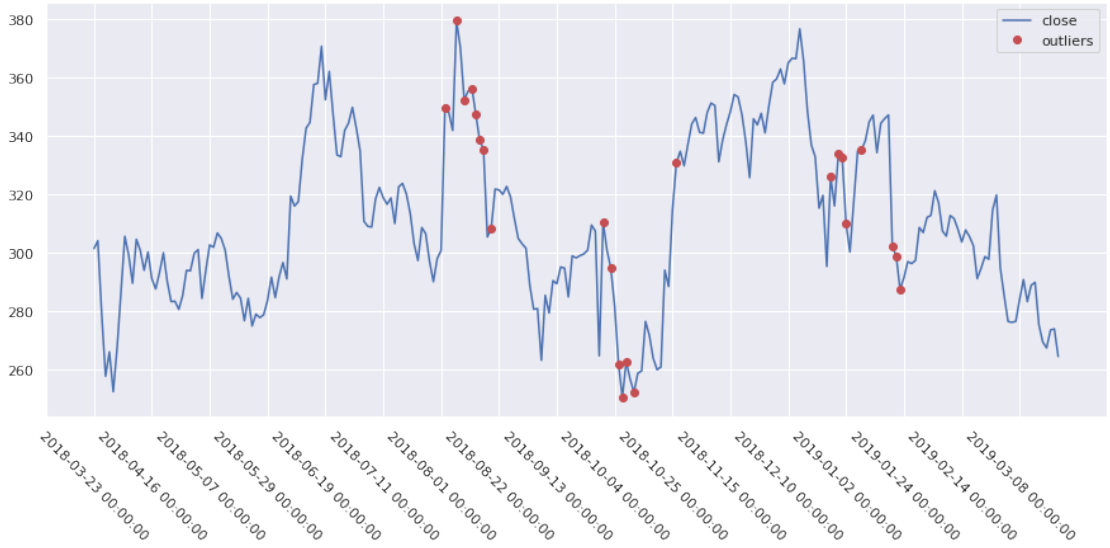
[36]:
```
from sklearn.svm import OneClassSVM

X = df_crosscorrelated.iloc[:,1:].dropna().values
np_scaled = StandardScaler().fit_transform(X)
model = OneClassSVM(nu=outliers_fraction, kernel="rbf", gamma=0.01)
model.fit(np_scaled)
outliers = model.predict(np_scaled)

plt.figure(figsize=(15, 6))
plt.plot(df_crosscorrelated['Close'], label='close',c='b')
plt.plot(df_crosscorrelated['Close'], 'o', label='outliers',
         markevery=(np.where(outliers==-1)[0] + ori_len).tolist(),c='r')
plt.xticks(np.arange(df_crosscorrelated.shape[0])[::
 ↪15],df_crosscorrelated['Date'][::15],rotation='-45')
plt.legend()
plt.show()
```



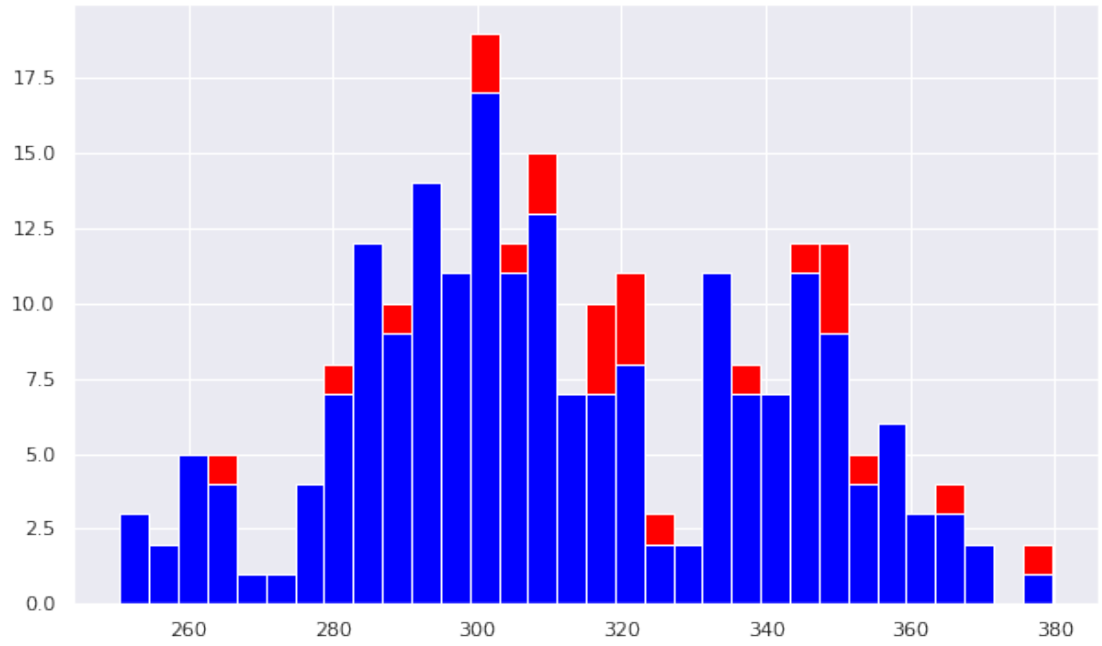
[37]:
```
close = df_crosscorrelated['Close'].values
a = close[np.where(outliers==1)[0]]
b = close[np.where(outliers==-1)[0]]

fig, axs = plt.subplots(figsize=(10,6))
axs.hist([a,b], bins=32, stacked=True, color=['blue', 'red'])
plt.show()
```



[38]:
```
from sklearn.covariance import EllipticEnvelope

envelope = EllipticEnvelope(contamination = outliers_fraction)
X = df_crosscorrelated.iloc[:,1:].dropna().values
np_scaled = StandardScaler().fit_transform(X)
envelope.fit(np_scaled)
outliers = envelope.predict(np_scaled)

plt.figure(figsize=(15, 6))
plt.plot(df_crosscorrelated['Close'], label='close',c='b')
plt.plot(df_crosscorrelated['Close'], 'o', label='outliers',
         markevery=(np.where(outliers==-1)[0] + ori_len).tolist(),c='r')
plt.xticks(np.arange(df_crosscorrelated.shape[0])[::
 ↪15],df_crosscorrelated['Date'][::15],rotation='-45')
plt.legend()
plt.show()
```



[39]:
```
close = df_crosscorrelated['Close'].values
a = close[np.where(outliers==1)[0]]
b = close[np.where(outliers==-1)[0]]

fig, axs = plt.subplots(figsize=(10,6))
axs.hist([a,b], bins=32, stacked=True, color=['blue', 'red'])
plt.show()
```


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
260
280
300
320
340
360
380

[ ]: