Carga Librerias

In [1]:
```
# Librerias
import warnings
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import plotly.express as px

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.decomposition import PCA

from sklearn import svm
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error, confusion_matrix, ConfusionMatri
```

In [2]:
```
def plot_metrics(model, x, y, pred_y):
    print('R^2 score:\t', r2_score(y, pred_y))
    print('MSE:\t\t', mean_squared_error(y, pred_y))
    print('RMSE:\t\t', mean_squared_error(y, pred_y, squared=False))
    print('MAE:\t\t', mean_absolute_error(y, pred_y))

    fig, ax = plt.subplots(1)
    matplotlib.rc('figure', figsize=(15, 5))
    ax.plot(y)
    ax.plot(pred_y)
```

Carga Datos

In [3]:
```
DATA_DIR = "C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets"

train_data_cut = pd.read_csv(f"{DATA_DIR}/train_data.csv")
test_data_cut = pd.read_csv(f"{DATA_DIR}/test_data.csv")

train_data_uncut = pd.read_csv(f"{DATA_DIR}/train_data_uncut.csv")
test_data_uncut = pd.read_csv(f"{DATA_DIR}/test_data_uncut.csv")

# X_train = pd.read_csv(f"{DATA_DIR}/X_train.csv")
# y_train = pd.read_csv(f"{DATA_DIR}/y_train.csv")
# X_test = pd.read_csv(f"{DATA_DIR}/X_test.csv")
# y_test = pd.read_csv(f"{DATA_DIR}/y_test.csv")
```

In [4]:
```
data_cut = pd.concat([train_data_cut, test_data_cut])
data_uncut = pd.concat([train_data_uncut, test_data_uncut])

features = ['volt', 'rotate', 'pressure', 'vibration', 'error1', 'error2', 'error3',
            'error4', 'error5', 'volt_3h_mean', 'rotate_3h_mean',
            'pressure_3h_mean', 'vibration_3h_mean', 'volt_24h_mean',
            'rotate_24h_mean', 'pressure_24h_mean', 'vibration_24h_mean',
            'error1_count', 'error2_count', 'error3_count', 'error4_count',
            'error5_count']
label = ['RUL']

data_cut = data_cut[features+label]
data_uncut = data_uncut[features+label]
```

Normalización de datos MinMax

In [5]:
```
feature_scaler_cut = MinMaxScaler(feature_range=(0,1))
label_scaler_cut = MinMaxScaler(feature_range=(0,1))

feature_scaler_cut.fit(data_cut[features])
label_scaler_cut.fit(data_cut[label].values.reshape(-1,1))
```

Out[5]:
```
MinMaxScaler()
```

In [6]:
```
feature_scaler_uncut = MinMaxScaler(feature_range=(0,1))
label_scaler_uncut = MinMaxScaler(feature_range=(0,1))
```

```
feature_scaler_uncut.fit(data_uncut[features])
label_scaler_uncut.fit(data_uncut[label].values.reshape(-1,1))
```

Out[6]:
```
MinMaxScaler()
```

In [7]:
```
data_norm_cut = data_cut.copy()
data_norm_cut[features] = feature_scaler_cut.transform(data_cut[features])
data_norm_cut[label] = label_scaler_cut.transform(data_cut[label].values.reshape(-1,1))

data_norm_uncut = data_uncut.copy()
data_norm_uncut[features] = feature_scaler_uncut.transform(data_uncut[features])
data_norm_uncut[label] = label_scaler_uncut.transform(data_uncut[label].values.reshape(-1,1))
```

Dataset Train/Test

In [8]:
```
X_train_cut, X_test_cut, y_train_cut, y_test_cut = train_test_split(data_norm_cut[features], data_norm_cut[labe
```

In [9]:
```
X_train_uncut, X_test_uncut, y_train_uncut, y_test_uncut = train_test_split(data_norm_uncut[features], data_nor
```

Control/Swap de variables

Esta celda sirve para pasar de los datos con ciclos homogenizados al conjunto entero con todos los ciclos sin homogenizar

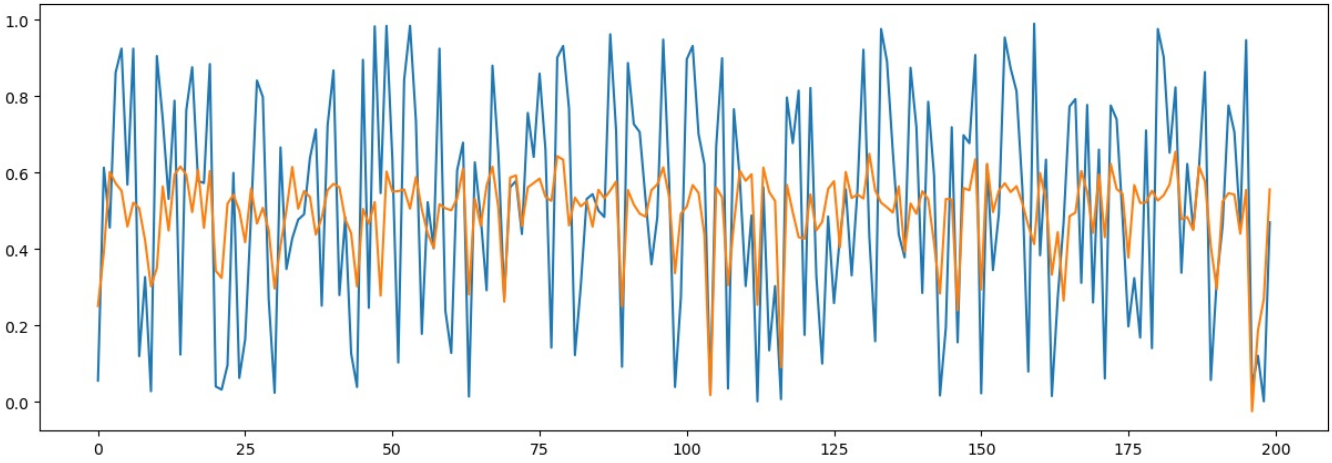
In [10]:
```
# Cut cycles
X_train = X_train_cut
y_train = y_train_cut
X_test = X_test_cut
y_test = y_test_cut

# Uncut cycles
# X_train = X_train_uncut
# y_train = y_train_uncut
# X_test = X_test_uncut
# y_test = y_test_uncut
```

Linear Regression

In [11]:
```
model_lr = LinearRegression().fit(X_train, y_train)
pred_lr = model_lr.predict(X_test)
```

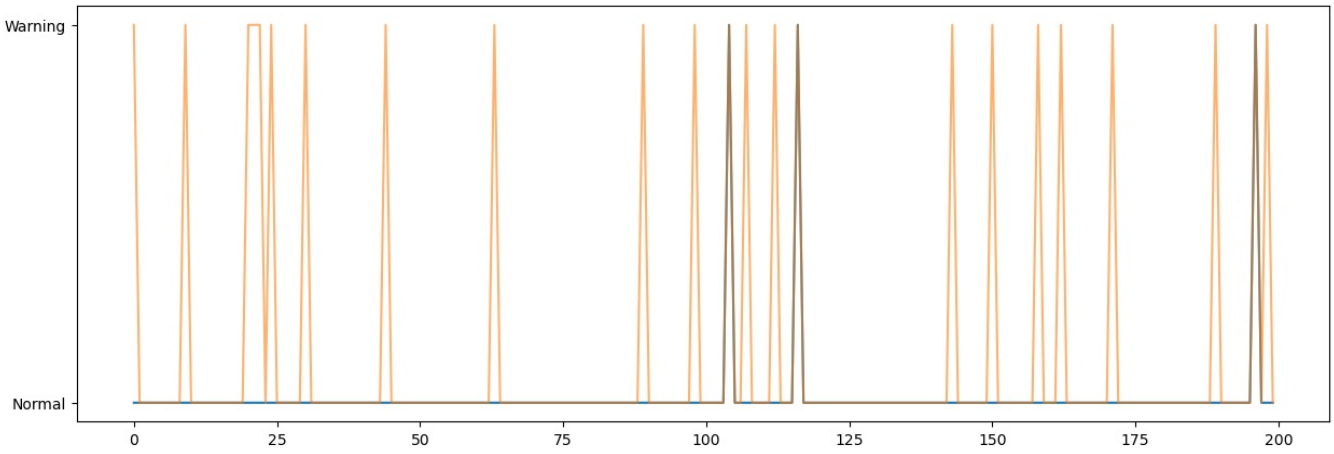
In [31]:
```
plot_metrics(model_lr, X_test[:200], y_test[:200].values, pred_lr[:200])
```

```
R^2 score:       0.2588718134999083
MSE:             0.06381095593418949
RMSE:            0.25260830535473194
MAE:             0.21520199012181682
```



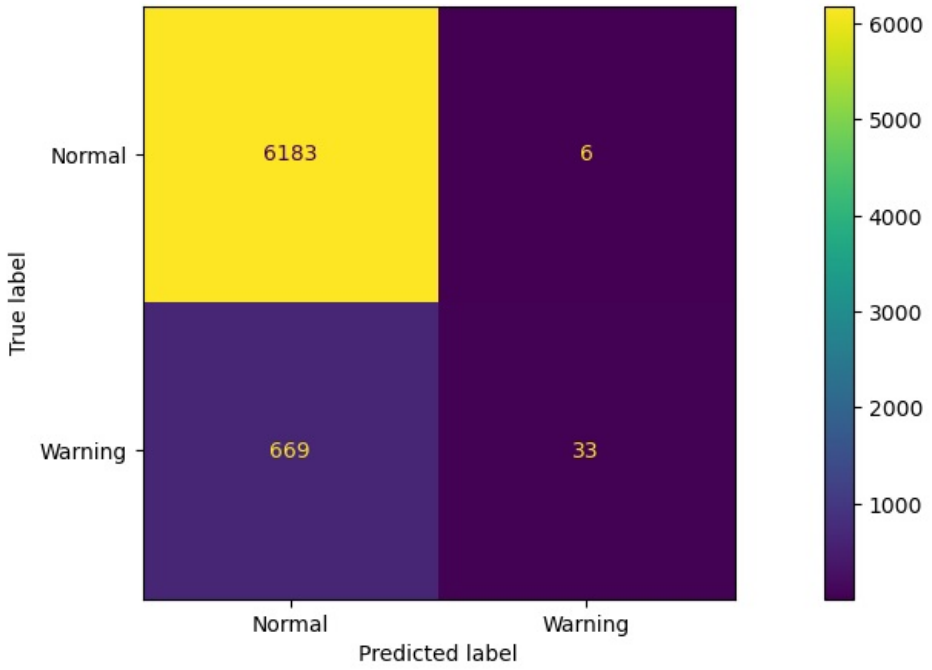
In [13]:
```
pred_lr_inv = label_scaler_cut.inverse_transform(pred_lr.reshape(-1, 1))
y_test_cut_inv = label_scaler_cut.inverse_transform(y_test_cut.values.reshape(-1, 1))

y_test_cut_inv_class = ['Normal' if x >= 72 else 'Warning' for x in y_test_cut_inv]
pred_lr_inv_class = ['Normal' if x >= 72 else 'Warning' for x in pred_lr_inv]

plt.plot(pred_lr_inv_class[:200])
plt.plot(y_test_cut_inv_class[:200], alpha = 0.6)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x1f1575c26a0>]
```



In [14]:
```
lr_cm = confusion_matrix(y_test_cut_inv_class, pred_lr_inv_class)

print('Accuracy: ', accuracy_score(y_test_cut_inv_class, pred_lr_inv_class))
ConfusionMatrixDisplay(lr_cm, display_labels=['Normal', 'Warning']).plot()
```

```
Accuracy:  0.9020461471484545
```
Out[14]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f155a27640>
```

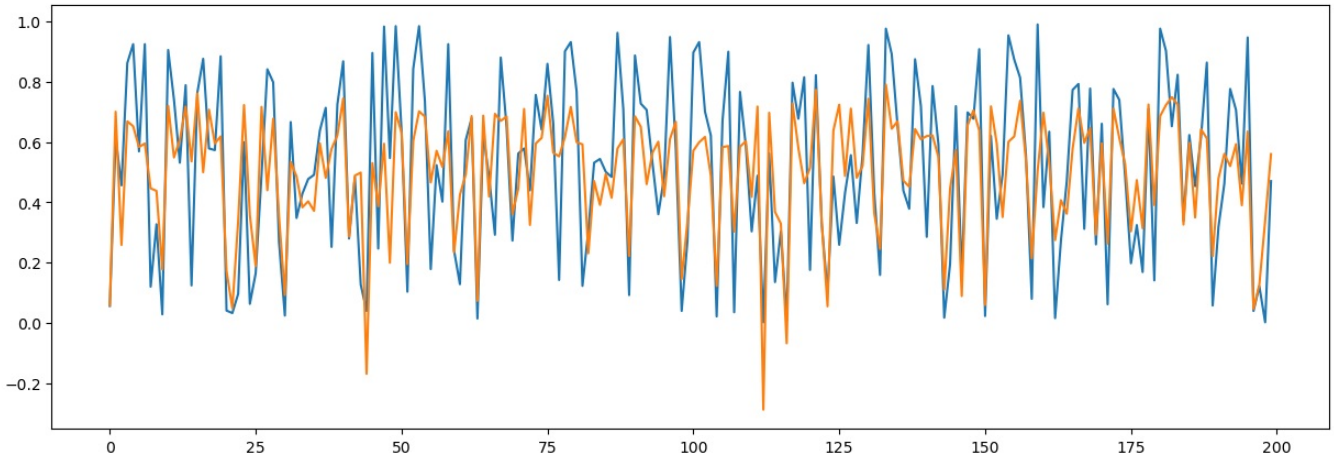


SVM Regressor

In [15]:
```
model_svm = svm.SVR().fit(X_train, y_train.to_numpy().ravel())
pred_svm = model_svm.predict(X_test)
```

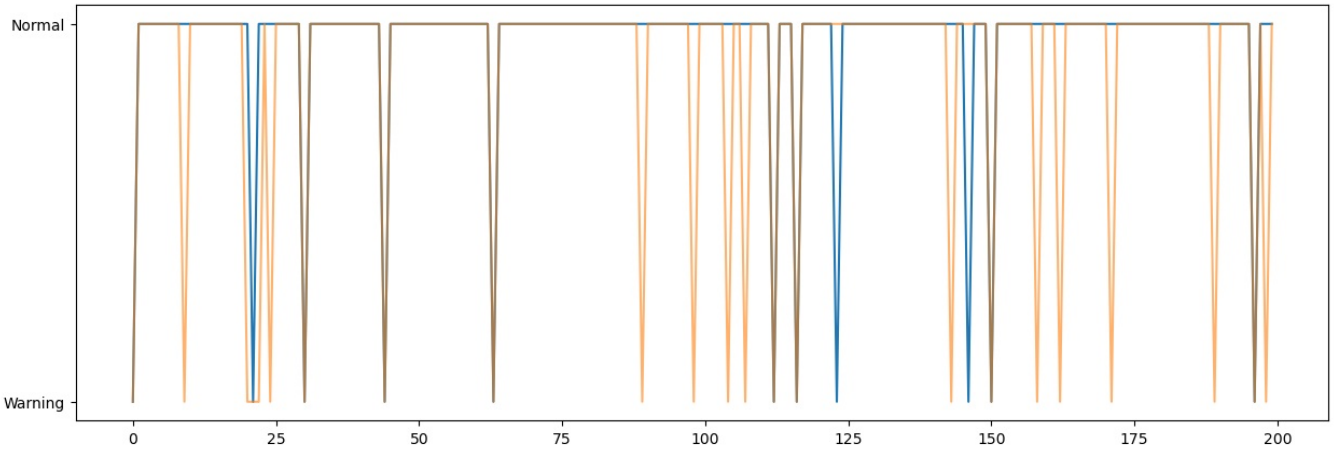
In [16]:
```
plot_metrics(model_svm, X_test[:200], y_test[:200].values, pred_svm[:200])
```

```
R^2 score:       0.5657667864012093
MSE:             0.03738737368627998
RMSE:            0.1933581487454821
MAE:             0.15527680953807102
```



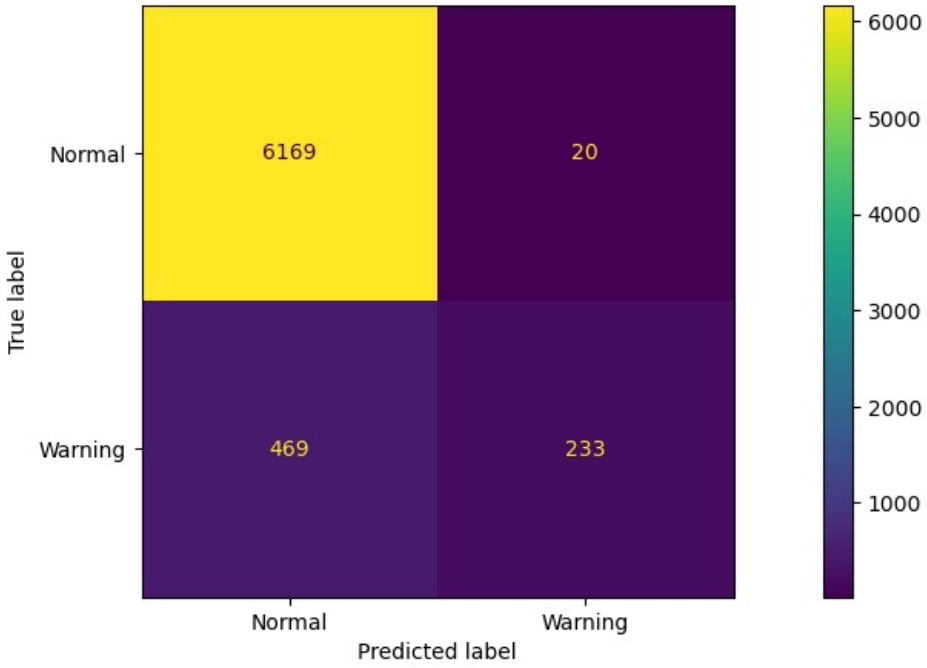
In [17]:
```
pred_svm_inv = label_scaler_cut.inverse_transform(pred_svm.reshape(-1, 1))
y_test_cut_inv = label_scaler_cut.inverse_transform(y_test_cut.values.reshape(-1, 1))

y_test_cut_inv_class = ['Normal' if x >= 72 else 'Warning' for x in y_test_cut_inv]
pred_svm_inv_class = ['Normal' if x >= 72 else 'Warning' for x in pred_svm_inv]


plt.plot(pred_svm_inv_class[:200])
plt.plot(y_test_cut_inv_class[:200], alpha = 0.6)
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x1f1582bbca0>]
```



In [18]:
```
svm_cm = confusion_matrix(y_test_cut_inv_class, pred_svm_inv_class)

print('Accuracy: ', accuracy_score(y_test_cut_inv_class, pred_svm_inv_class))
ConfusionMatrixDisplay(svm_cm, display_labels=['Normal', 'Warning']).plot()
```

Out[18]:
```
Accuracy:  0.9290378754897692
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f157a3a580>
```

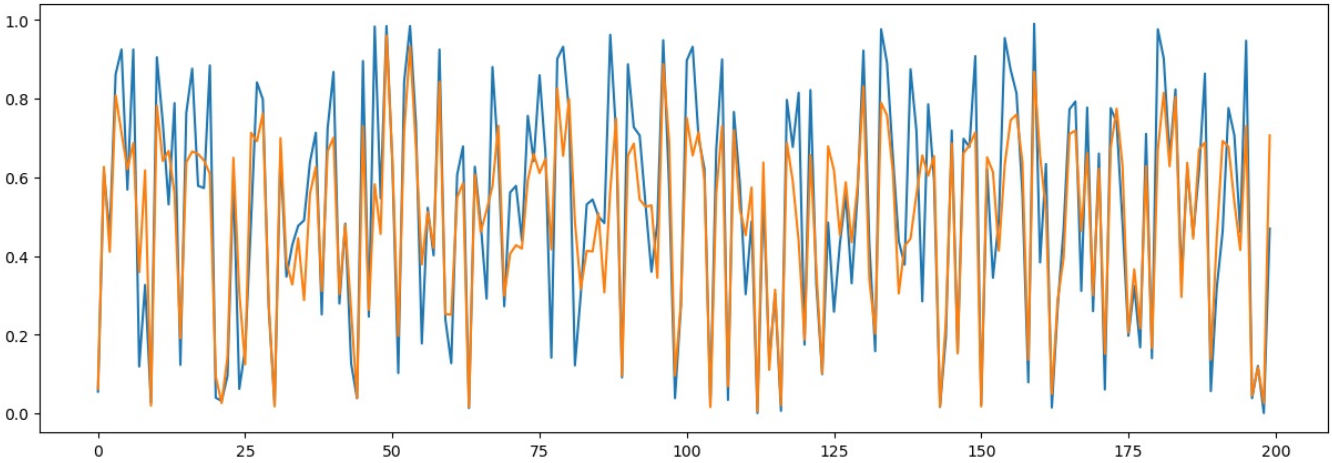


Random Forest Regressor

In [19]:
```
model_rf = RandomForestRegressor().fit(X_train, y_train.to_numpy().ravel())
pred_rf = model_rf.predict(X_test)
```

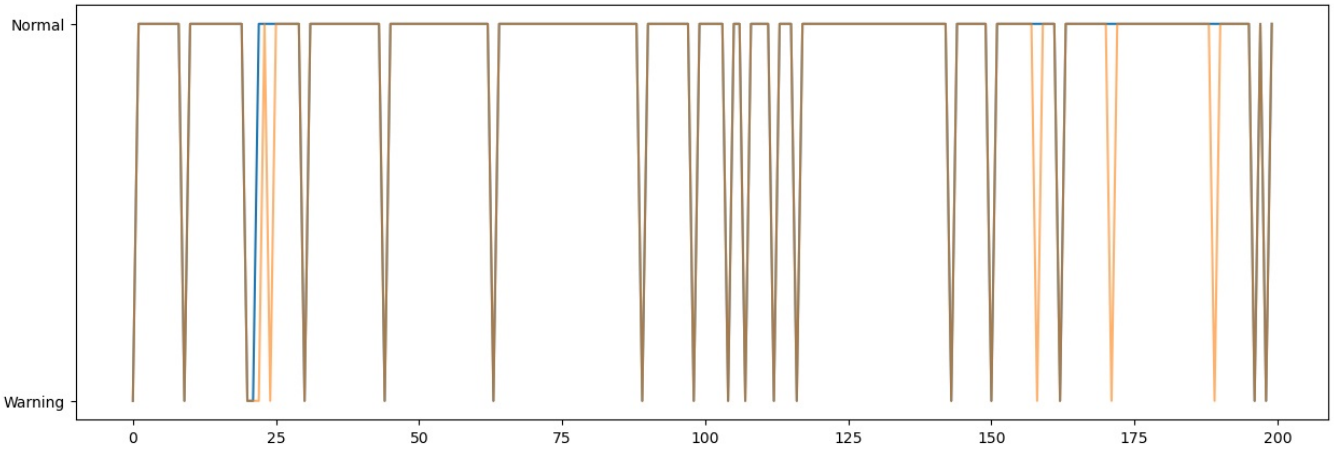
In [20]:
```
plot_metrics(model_rf, X_test[:200], y_test[:200].values, pred_rf[:200])
```

```
R^2 score:       0.7752798899252249
MSE:             0.0193483466189906
RMSE:            0.1390983343501661
MAE:             0.101028859527121
```



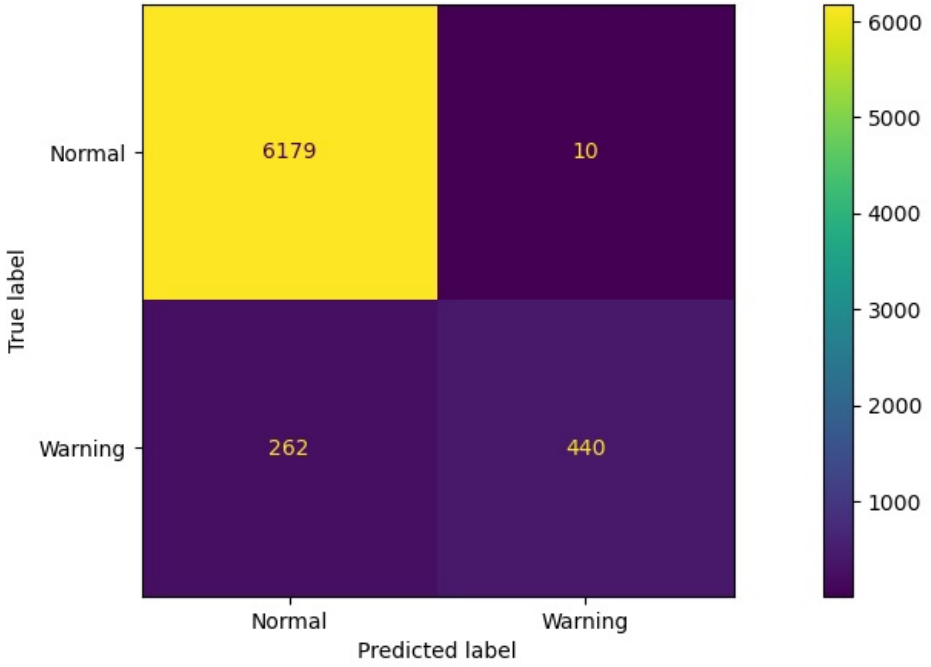
In [21]:
```
pred_rf_inv = label_scaler_cut.inverse_transform(pred_rf.reshape(-1, 1))
y_test_cut_inv = label_scaler_cut.inverse_transform(y_test_cut.values.reshape(-1, 1))

y_test_cut_inv_class = ['Normal' if x >= 72 else 'Warning' for x in y_test_cut_inv]
pred_rf_inv_class = ['Normal' if x >= 72 else 'Warning' for x in pred_rf_inv]


plt.plot(pred_rf_inv_class[:200])
plt.plot(y_test_cut_inv_class[:200], alpha = 0.6)
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x1f158107400>]
```



In [22]:
```
rf_cm = confusion_matrix(y_test_cut_inv_class, pred_rf_inv_class)

print('Accuracy: ', accuracy_score(y_test_cut_inv_class, pred_rf_inv_class))
ConfusionMatrixDisplay(rf_cm, display_labels=['Normal', 'Warning']).plot()
```

Out[22]:
```
Accuracy:  0.9605282252213031
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f157e9c6a0>
```

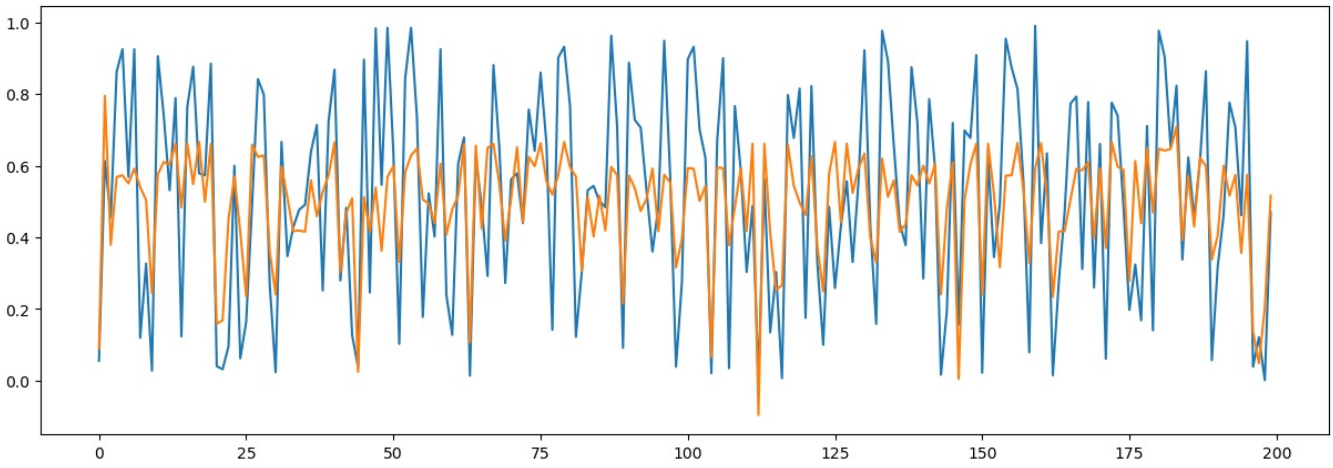


Gradient boosting Regressor

In [23]:
```
model_gb = GradientBoostingRegressor().fit(X_train, y_train.to_numpy().ravel())
pred_gb = model_gb.predict(X_test)
```

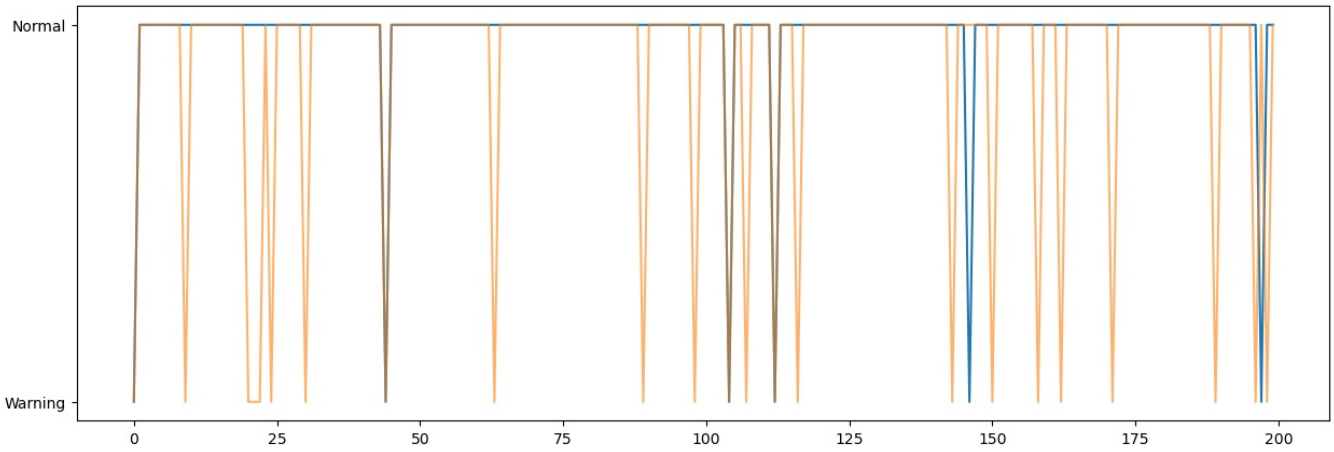
In [24]:
```
plot_metrics(model_gb, X_test[:200], y_test[:200].values, pred_gb[:200])
```

```
R^2 score:       0.481953454305322
MSE:             0.04460368111930944
RMSE:            0.21119583594216398
MAE:             0.17482294880465282
```



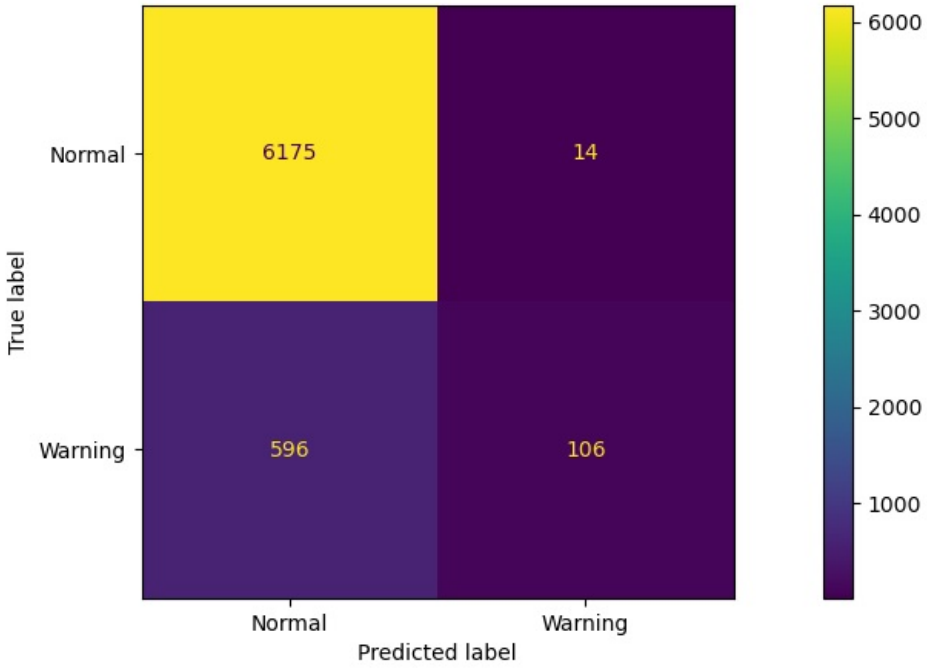
In [25]:
```
pred_gb_inv = label_scaler_cut.inverse_transform(pred_gb.reshape(-1, 1))
y_test_cut_inv = label_scaler_cut.inverse_transform(y_test_cut.values.reshape(-1, 1))

y_test_cut_inv_class = ['Normal' if x >= 72 else 'Warning' for x in y_test_cut_inv]
pred_gb_inv_class = ['Normal' if x >= 72 else 'Warning' for x in pred_gb_inv]


plt.plot(pred_gb_inv_class[:200])
plt.plot(y_test_cut_inv_class[:200], alpha = 0.6)
```

Out[25]:
```
[<matplotlib.lines.Line2D at 0x1f1589c2fa0>]
```



In [26]:
```
gb_cm = confusion_matrix(y_test_cut_inv_class, pred_gb_inv_class)

print('Accuracy: ', accuracy_score(y_test_cut_inv_class, pred_gb_inv_class))
ConfusionMatrixDisplay(gb_cm, display_labels=['Normal', 'Warning']).plot()
```

Out[26]:
```
Accuracy:  0.9114787403860107
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f15831a2e0>
```

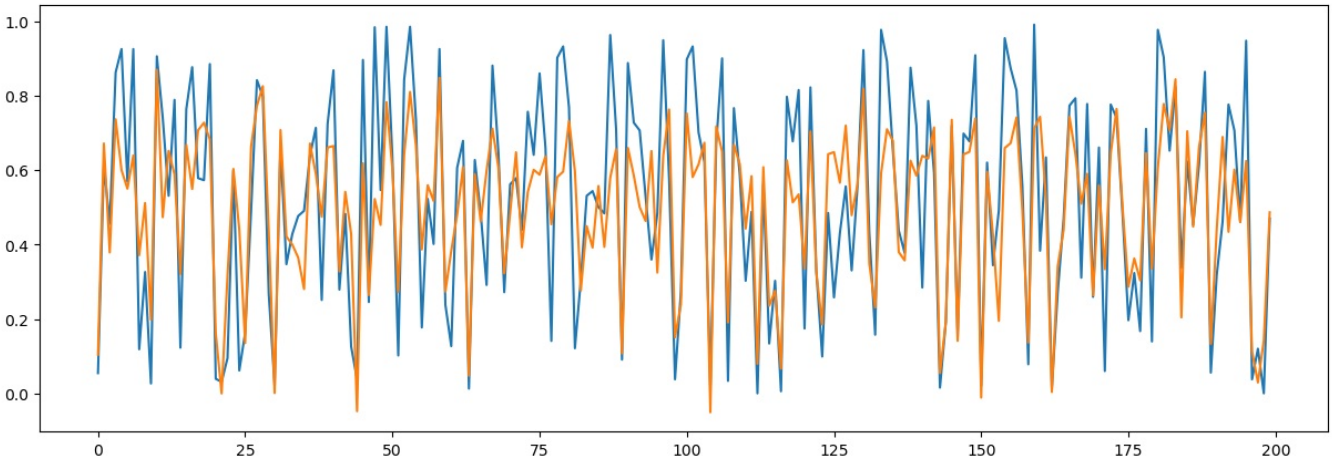


XGBoosting Regressor

In [27]:
```
model_xgb = XGBRegressor().fit(X_train, y_train.to_numpy().ravel())
pred_xgb = model_xgb.predict(X_test)
```

In [28]:
```
plot_metrics(model_xgb, X_test[:200], y_test.to_numpy().ravel()[:200], pred_xgb[:200])
```

```
R^2 score:       0.6682029106631473
MSE:             0.028567648393930875
RMSE:            0.1690196686599843
MAE:             0.1320614130136721
```

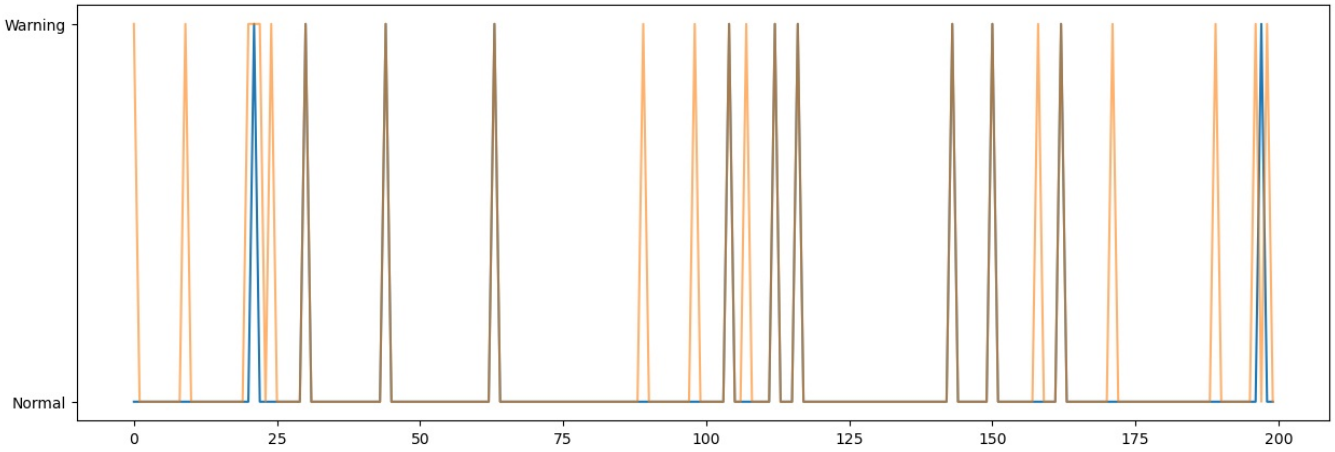


In [29]:
```
pred_xgb_inv = label_scaler_cut.inverse_transform(pred_xgb.reshape(-1, 1))
y_test_cut_inv = label_scaler_cut.inverse_transform(y_test_cut.values.reshape(-1, 1))

y_test_cut_inv_class = ['Normal' if x >= 72 else 'Warning' for x in y_test_cut_inv]
pred_xgb_inv_class = ['Normal' if x >= 72 else 'Warning' for x in pred_xgb_inv]


plt.plot(pred_xgb_inv_class[:200])
```

```
plt.plot(y_test_cut_inv_class[:200], alpha = 0.6)
```

Out[29]:

```
[<matplotlib.lines.Line2D at 0x1f15bea7af0>]
```



In [30]:

```
xgb_cm = confusion_matrix(y_test_cut_inv_class, pred_xgb_inv_class)

print('Accuracy: ', accuracy_score(y_test_cut_inv_class, pred_xgb_inv_class))
ConfusionMatrixDisplay(xgb_cm, display_labels=['Normal', 'Warning']).plot()
```

```
Accuracy:  0.9377448846321289
```

Out[30]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f15bebd040>
```