In [6]:

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_openml
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split

import time


seed = 1234
np.random.seed(seed)

X, y = fetch_openml('Fashion-MNIST', as_frame=False, return_X_y=True)
y = y.astype(int)

X_tr,X_te,y_tr,y_te = train_test_split(X,y,test_size= 0.3, random_state=seed,shuffle=True)
```

## Time used for fit and predict

In [7]:

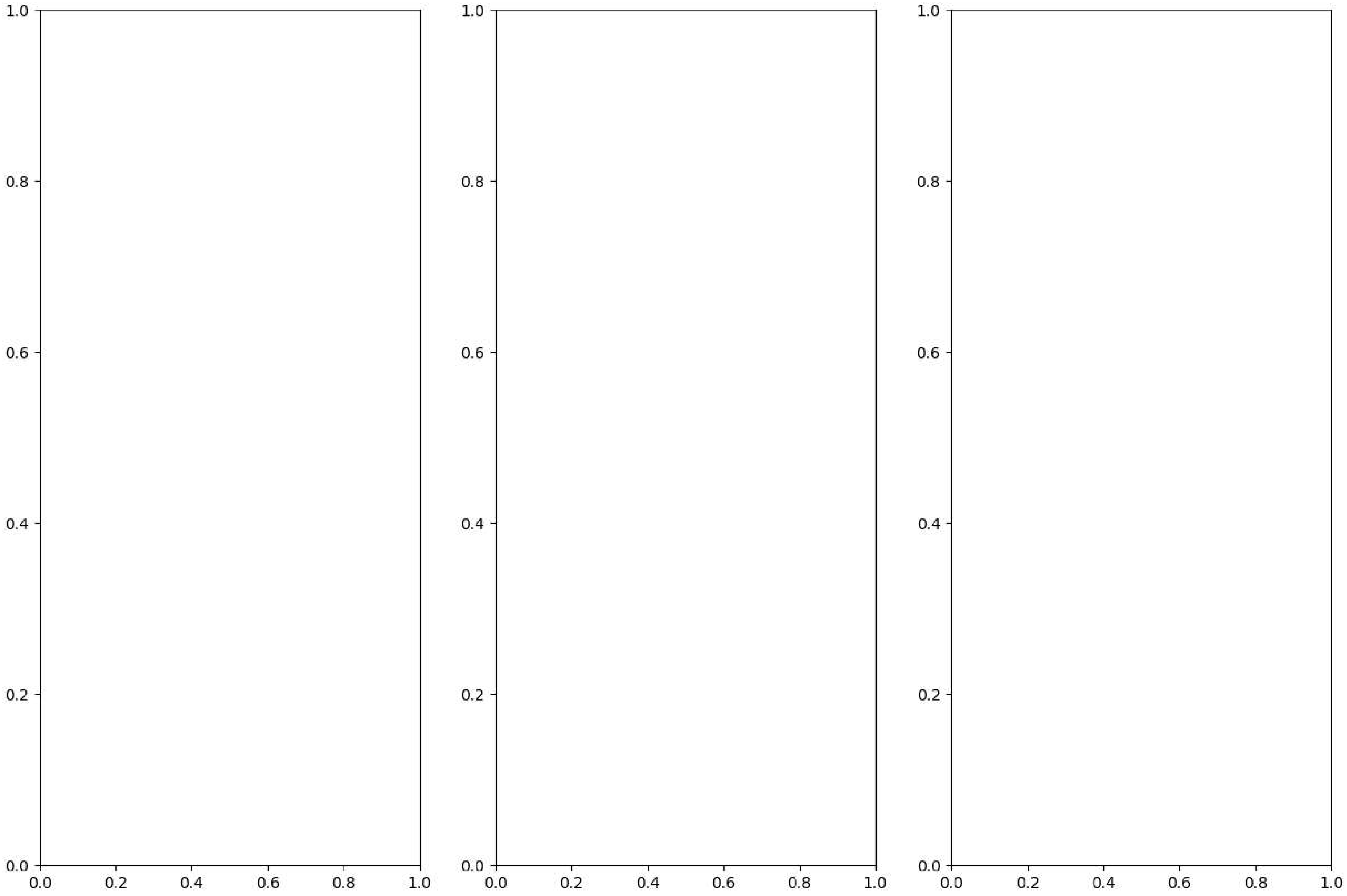
```
randomForest = RandomForestClassifier(random_state=seed)
start_fit = time.time()
randomForest.fit(X_tr,y_tr)
end_fit = time.time()

start_predict = time.time()
rf_predict = randomForest.predict(X_te)
end_predict = time.time()

time_fit = end_fit-start_fit
time_predict = end_predict - start_predict

print(f"time for random forest to fit fashion MNIST: {time_fit}")
print(f"time for random forest to predict fashion MNIST: {time_predict}")
```

```
time for random forest to fit fashion MNIST: 57.349565267562866
time for random forest to predict fashion MNIST: 0.6680629253387451
```

## Error rate with Complexity of Random Forest

In [8]:

```
# change the number of estimator, max number of tree depth
# The hyperparameters for random forests include the number of trees M, the number of randomly selected features R at each node,
# min-samples). Random forests tend not to be particularly sensitive to the settings of these parameters and
# default values tend to generally work well

# turn on and off the boostraping, set max_depth or not to see the impact
# with gini index, max_features = sqrt(d)

n_tree = [10, 50, 100, 200, 400, 800]


err_tr = np.zeros(len(n_tree))
err_te = np.zeros(len(n_tree))

err_tr_b = np.zeros(len(n_tree))
err_te_b = np.zeros(len(n_tree))

err_tr_d = np.zeros(len(n_tree))
err_te_d = np.zeros(len(n_tree))
for i,n in enumerate(n_tree):
    # regular with default settings other than number of estimators
    R_randomForest = RandomForestClassifier(n_estimators=n,random_state=seed)
    R_randomForest.fit(X_tr,y_tr)
    err_tr[i] = 1. - R_randomForest.score(X_tr, y_tr)
    err_te[i] = 1. - R_randomForest.score(X_te, y_te)

    # no boostraping
    nB_randomForest = RandomForestClassifier(n_estimators=n,random_state=seed,bootstrap = False)
    nB_randomForest.fit(X_tr,y_tr)
    err_tr_b[i] = 1. - nB_randomForest.score(X_tr, y_tr)
    err_te_b[i] = 1. - nB_randomForest.score(X_te, y_te)

    # with set max_depth
    m_randomForest = RandomForestClassifier(n_estimators=n,random_state=seed,max_depth=5)
    m_randomForest.fit(X_tr,y_tr)
    err_tr_d[i] = 1. - m_randomForest.score(X_tr, y_tr)
    err_te_d[i] = 1. - m_randomForest.score(X_te, y_te)
```

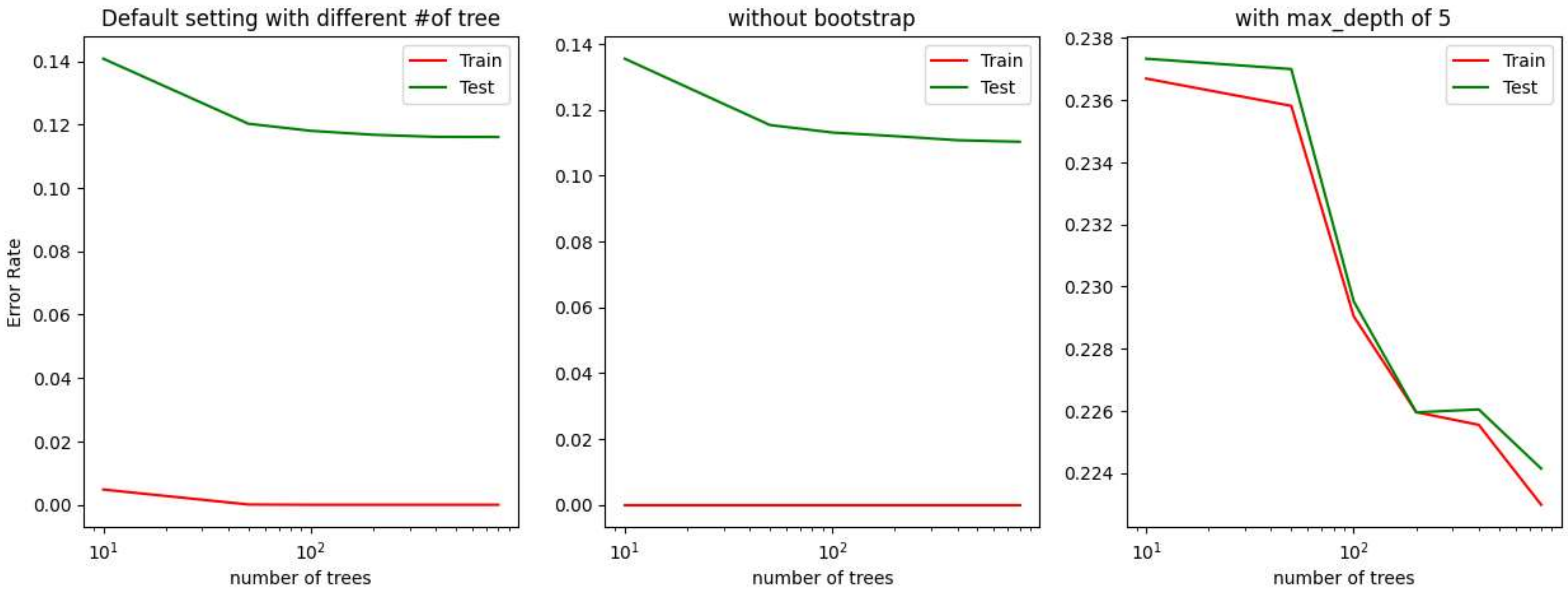

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [19]:

```
figure, axes = plt.subplots(1, 3 ,figsize=(15, 5))
axes[0].semilogx(n_tree,err_tr, c='red', label='Train')
axes[0].semilogx(n_tree,err_te, c='green', label='Test')
axes[0].set_title(f"Default setting with different #of tree")

axes[1].semilogx(n_tree,err_tr_b, c='red', label='Train')
axes[1].semilogx(n_tree,err_te_b, c='green', label='Test')
axes[1].set_title(f"without bootstrap")

axes[2].semilogx(n_tree,err_tr_d, c='red', label='Train')
axes[2].semilogx(n_tree,err_te_d, c='green', label='Test')
axes[2].set_title(f"with max_depth of 5")


axes[0].set_xlabel('number of trees')
axes[0].set_ylabel('Error Rate')
axes[0].legend()
axes[1].set_xlabel('number of trees')
axes[1].legend()
axes[2].set_xlabel('number of trees')
axes[2].legend()
```

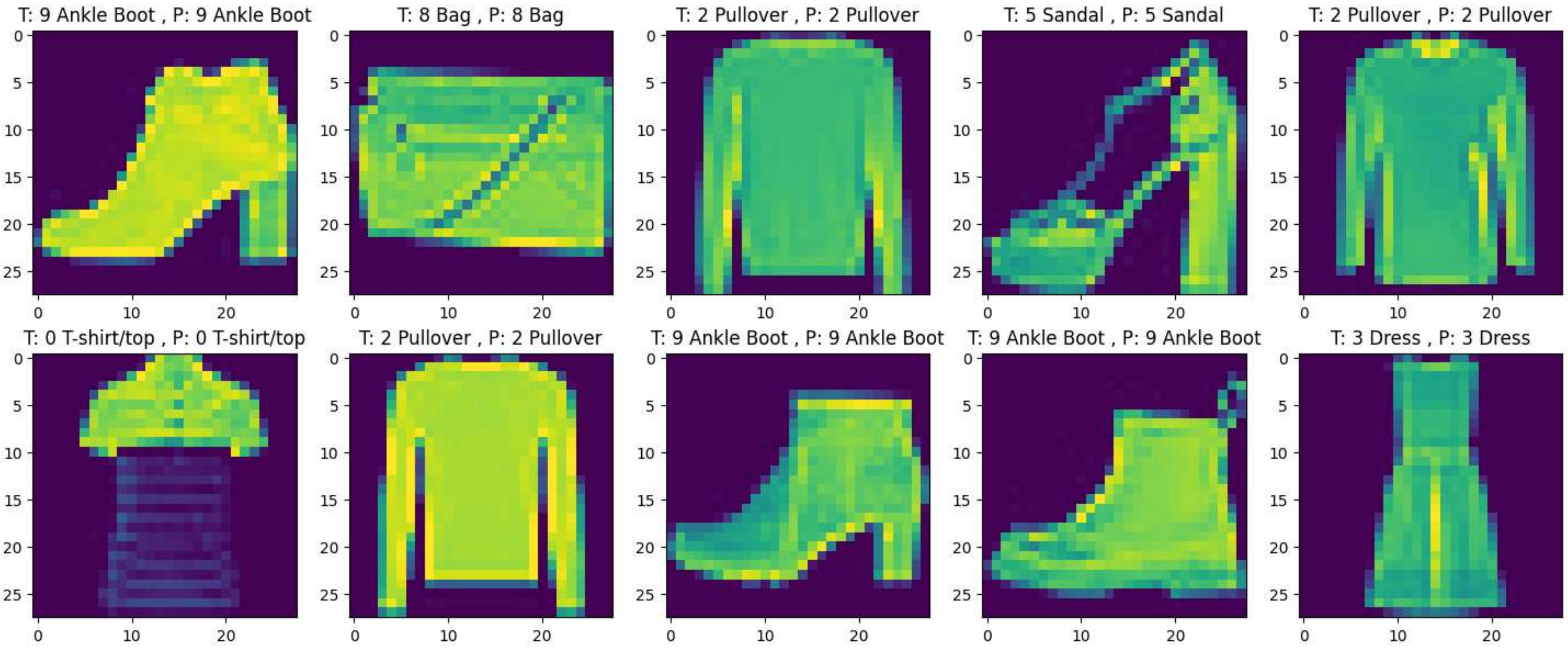
Out[19]:

```
<matplotlib.legend.Legend at 0x15200bb5900>
```



In [16]:

```
# do some visualization
classes = ('T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle Boot')
figure, axes = plt.subplots(2, 5 ,figsize=(18, 7))


for i in range(10):
    index = i+300
    img = X_te[index].reshape(28, 28)
    label = classes[y_te[index]]
    predict = classes[rf_predict[index]]
    axes[i//5,i%5].imshow(img)
    axes[i//5, i%5].set_title(f'T: {y_te[index]} {label} , P: {rf_predict[index]} {predict}')


plt.show()
```



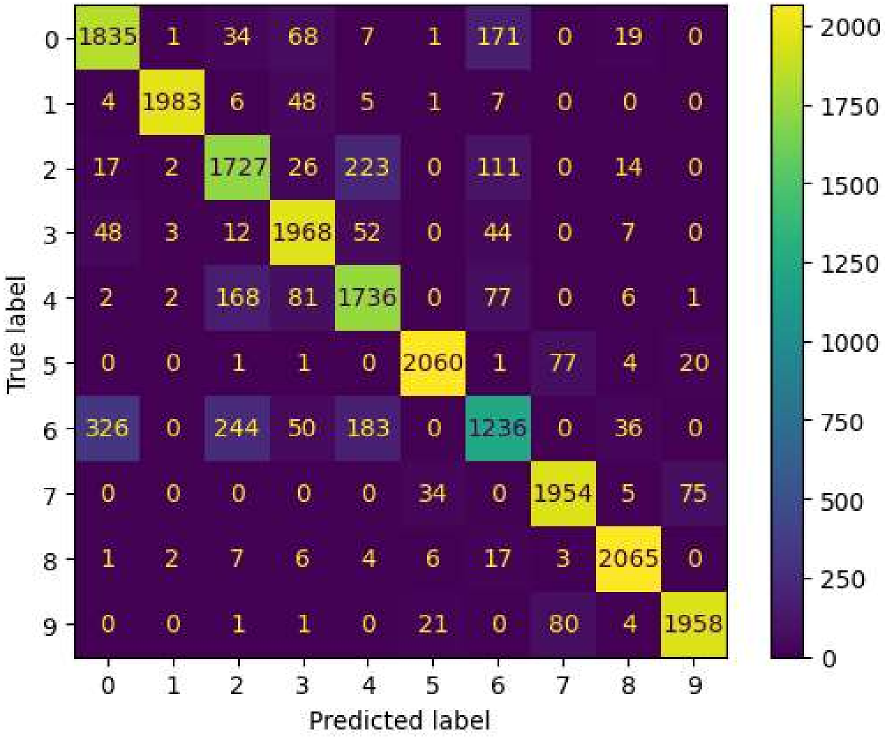
## Confusion Matrix

In [17]:

```
cm = confusion_matrix(y_te,rf_predict)
cm_disp = ConfusionMatrixDisplay(confusion_matrix = cm)
cm_disp.plot()
```

Out[17]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x152b85ae470>
```



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1835 | 1 | 34 | 68 | 7 | 1 | 171 | 0 | 19 | 0 |
| 1 | 4 | 1983 | 6 | 48 | 5 | 1 | 7 | 0 | 0 | 0 |
| 2 | 17 | 2 | 1727 | 26 | 223 | 0 | 111 | 0 | 14 | 0 |
| 3 | 48 | 3 | 12 | 1968 | 52 | 0 | 44 | 0 | 7 | 0 |
| 4 | 2 | 2 | 168 | 81 | 1736 | 0 | 77 | 0 | 6 | 1 |
| 5 | 0 | 0 | 1 | 1 | 0 | 2060 | 1 | 77 | 4 | 20 |
| 6 | 326 | 0 | 244 | 50 | 183 | 0 | 1236 | 0 | 36 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 34 | 0 | 1954 | 5 | 75 |
| 8 | 1 | 2 | 7 | 6 | 4 | 6 | 17 | 3 | 2065 | 0 |
| 9 | 0 | 0 | 1 | 1 | 0 | 21 | 0 | 80 | 4 | 1958 |

## Compare with Tree

In [18]:

```
from sklearn import tree
decisionTree = tree.DecisionTreeClassifier(random_state=seed)
decisionTree.fit(X_tr, y_tr)

error_rf = 1- randomForest.score(X_te,y_te)
error_dt = 1- decisionTree.score(X_te,y_te)

print(f"error rate for random forest with default settings on Fashion_MNIST: {error_rf}")
print(f"error rate for decision tree with default settings on Fashion_MNIST: {error_dt}")
```

```
error rate for random forest with default settings on Fashion_MNIST: 0.118
error rate for decision tree with default settings on Fashion_MNIST: 0.20971428571428574
```