In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

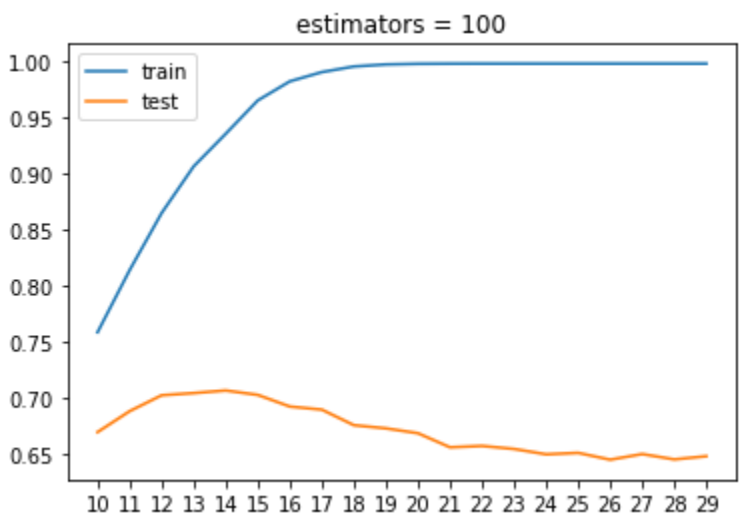
In [ ]:
```
df_ori = pd.read_csv('data/gdf_final.csv').dropna()
X_col = [
    'dist',
    'delta_time',
    'trip_hour',
    'avgtemp',
    'population_16_with_earnings',
    'median_earnings_(dollars)',
    'median_age_(years)'
]
y_col = ['usage_counts']
classes = ['low', 'mid', 'high']
X = StandardScaler().fit_transform(df_ori[X_col])
```

In [ ]:
```
from sklearn.ensemble import GradientBoostingRegressor
y = df_ori[y_col].to_numpy().flatten()
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
regr = GradientBoostingRegressor(criterion='squared_error',
                                 n_estimators=100,
                                 max_features='sqrt',
                                 max_depth=10,
                                 random_state=0)
regr.fit(X_train, y_train)
regr.score(X_train, y_train), regr.score(X_test, y_test)
```

Out[ ]:
```
(0.7582364449374186, 0.6691779638406311)
```

In [ ]:
```
n = 20
n_estimators = 100
depthList = [i+10 for i in range(n)]
train_score_list = np.zeros(n)
test_score_list = np.zeros(n)
for idx, d in enumerate(depthList):
    regr = GradientBoostingRegressor(criterion='squared_error',
                                     n_estimators=n_estimators,
                                     max_features='sqrt',
                                     max_depth=d,
                                     random_state=0)
    regr.fit(X_train, y_train)
    train_score_list[idx] = regr.score(X_train, y_train)
    test_score_list[idx] = regr.score(X_test, y_test)
plt.plot(list(range(n)), train_score_list, label='train')
plt.plot(list(range(n)), test_score_list, label='test')
plt.xticks(list(range(n)), depthList)
plt.title('estimators = 100')
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x26800e2dd60>
```



The best depth is 14

In [ ]:
```
depth = depthList[np.argmax(test_score_list)]
regr = GradientBoostingRegressor(max_depth=depth,
                                 n_estimators=n_estimators,
                                 criterion='squared_error',
                                 max_features='sqrt',
                                 random_state=0)
regr.fit(X_train, y_train)
regr.score(X_train, y_train), regr.score(X_test, y_test)
```

Out[ ]:
```
(0.9352177852113234, 0.7062878593191781)
```

In [ ]:
```

```