수상작리뷰보고서(12/16)

태그

[풍속 예측 AI 해커톤]

https://dacon.io/competitions/official/236126/overview/description

데이터

1. ID
2. Month
3. Day
4. Measurement Time
5. T(C): 측정된 기온 섭씨
6. T(K): 측정 기온을 절대 온도(켈빈, k)로 변환
7. Dew T: 이슬점 온도
8. Relative Humidity: 상대 습도
9. P(mbar): 대기압
10. Saturation Vapor P (mbar): 공기 포화 상태에 있을 때 수증기압
11. Actual Vapor P (mbar): 공기 중 실제 수증기압
12. Vapor Content: 공기 1kg당 포함된 수증기량(g)
13. Vapor P Shortage (mbar): 포화 수증기압과 실제 수증가압의 차이
14. Air Density: 공기의 밀도
15. Direction (deg): 풍향

코드 흐름

1. 전처리: 주기성을 띄는 feature에 대해서는 sin, cos를 적용했고 범주형 feature는 target encoding을 진행

```
def preprocess(df, test=False) :

  if test :
    df.columns = ['ID', 'Month', 'Day', 'Measurement Time', '
    'Dew T(°C)', 'Relative Humidity (%)', 'P (mbar)', 'Satura
    'Actual Vapor P(mbar)', 'Vapor P Shortage (mbar)', 'Vapor
    'Direction (deg)']
  else :
    df.columns = ['ID', 'Month', 'Day', 'Measurement Time', '
      'Dew T(°C)', 'Relative Humidity (%)', 'P (mbar)', 'Satu
      'Actual Vapor P(mbar)', 'Vapor P Shortage (mbar)', 'Vap
      'Direction (deg)', 'Velocity (m/s)']

  df['Measurement Time'] = df['Measurement Time'].replace({'사
  df['Time_cos'] = df['Measurement Time'].apply(lambda x : np
  df['Time_sin'] = df['Measurement Time'].apply(lambda x : np

  df['Direction (sign)'] = ((df['Direction (deg)'] + 11.25) %
  df['Direction (sign)'] = df['Direction (sign)'].astype('int

  df['Date'] = df['Month'].apply(lambda x : format(x, '02')) +

  df['cat_Month'] = df['Month'].astype('category')
  df['cat_day'] = df['Day'].astype('category')
  df['cat_Measurement Time'] = df['Measurement Time'].astype(

  df['Direction_x'] = df['Direction (deg)'].apply(lambda x :
  df['Direction_y'] = df['Direction (deg)'].apply(lambda x :

  df['Month_cos'] = df['Month'].apply(lambda x : np.cos((x-1)
  df['Month_sin'] = df['Month'].apply(lambda x : np.sin((x-1)

  df = df.drop(['ID', 'T (K)', 'Vapor Content (g/kg)'], axis=

  return df
```

2. clutering

```
from pycaret import clustering
from sklearn.preprocessing import StandardScaler

# Cluster Setting

num_clusters = 3
clustering_features = ['Month_cos', 'Month_sin', 'T (°\u2063C

cluster = clustering.setup(data=df_train, ignore_features= [i
kmeans = clustering.create_model('kmeans', num_clusters=num_c

df_train['Cluster'] = clustering.predict_model(model = kmeans
df_train['Cluster'] = df_train['Cluster'].apply(lambda x : in

for i in range(num_clusters) :
  df_train[f'Cluster Distance {i}'] = kmeans.transform(cluste

df_train
```

3. setup

```
from pycaret.regression import *
import category_encoders

reg = setup(data=df_train, target='Velocity (m/s)', ignore_fe
reg.pipeline
best = compare_models(n_select=4)
```

4. Tuning

```
best[2] = tune_model(best[2], optimize='MAE', n_iter=100)
best[3] = tune_model(best[3], optimize='MAE', n_iter=100)
```

5. Stacking Ensemble, predict

```
stack_lr = stack_models(best, optimize='MAE', choose_better=T
stack_finalized = finalize_model(stack_lr)
df_test = preprocess(pd.read_csv('test.csv'), True)

df_test['Cluster'] = clustering.predict_model(model = kmeans,
df_test['Cluster'] = df_test['Cluster'].apply(lambda x : int(

for i in range(num_clusters) :
  df_test[f'Cluster Distance {i}'] = kmeans.transform(cluster

df_test

df_test = df_test.drop(['Direction_x', 'Direction_y'], axis=1
df_submit = pd.read_csv('sample_submission.csv')
df_submit['풍속 (m/s)'] = stack_finalized.predict(df_test)
df_submit
```

차별점, 배울점

1. PyCaret을 활용: 머신러닝 워크플로우를 간단하게 만들어주는 파이썬 라이브러리로, 다양한 모델 학습 및 튜닝을 지원함, 이러한 라이브러리를 이용해 ExtraTree, RandomForest, XGBoost, LGBM 등 다양한 모델을 선택 튜닝함.
2. Stack Ensemble을 이용: 여러 모델의 예측 결과를 결합하여 최종 예측을 생성하는 방식으로 모델 간의 장점을 결합하여 성능을 높였음.
3. 적절한 전처리: 범주형 데이터를 타겟 값과 연관된 값으로 인코딩하는 방식을 이용, 주기성을 가진 feature에 대해 sin, cos 변환을 적용하여 feature를 적절하게 전처리함.
4. 비슷한 feature를 묶는 클러스터링: 서로 비슷한 특성을 가진 feature를 묶어 추가 정보를 생성하거나, 차원을 축소하여 학습 성능을 향상