soc3

July 22, 2024

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

train = pd.read_csv('demand-forecasting-kernels-only/train.csv',␣
 ↪parse_dates=['date'])
maindataframe = train
maindataframe
```

[ ]:
```
             date  store  item  sales
0      2013-01-01      1     1     13
1      2013-01-02      1     1     11
2      2013-01-03      1     1     14
3      2013-01-04      1     1     13
4      2013-01-05      1     1     10
…             …    …    …    …
912995 2017-12-27     10    50     63
912996 2017-12-28     10    50     59
912997 2017-12-29     10    50     74
912998 2017-12-30     10    50     62
912999 2017-12-31     10    50     82

[913000 rows x 4 columns]
```

[ ]:
```
maindataframe.set_index('date', inplace=True)

import pandas as pd
import xgboost as xgb
from sklearn.metrics import root_mean_squared_error
import matplotlib.pyplot as plt
from datetime import timedelta


def filter_data(df, store_number, item_number):
    filtered_df = df[(df['store'] == store_number) & (df['item'] ==␣
 ↪item_number)]
    result_df = filtered_df[['sales']].copy()
    return result_df
```

```
def create_features(df):
    df['year'] = df.index.year
    df['month'] = df.index.month
    df['day'] = df.index.day
    df['day_of_week'] = df.index.dayofweek
    df['day_of_year'] = df.index.dayofyear
    df['week_of_year'] = df.index.isocalendar().week.astype(int)
    return df

def forecast_sales(df, store_number, item_number):
    result = filter_data(df, store_number, item_number)
    result = create_features(result)

    X = result.drop(['sales'], axis=1)
    y = result['sales']

    model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=10000,␣
 ↪learning_rate=0.01, early_stopping_rounds=50)
    model.fit(X, y, eval_set=[(X, y)], verbose=False)

    last_date = result.index[-1]
    future_dates = [last_date + timedelta(days=i) for i in range(1, 91)]  #␣
 ↪Next 3 months (approx. 90 days)

    future_df = pd.DataFrame(index=future_dates)
    future_df['year'] = future_df.index.year
    future_df['month'] = future_df.index.month
    future_df['day'] = future_df.index.day
    future_df['day_of_week'] = future_df.index.dayofweek
    future_df['day_of_year'] = future_df.index.dayofyear
    future_df['week_of_year'] = future_df.index.isocalendar().week.astype(int)

    X_future = future_df
    future_df['sales'] = model.predict(X_future)
    prediction_df = future_df[['sales']]

    # Visualization
    plt.figure(figsize=(15, 6))
    plt.plot(result.index, y, label='Actual Sales')
    plt.plot(result.index, model.predict(X), label='Predicted Sales')
    plt.plot(future_df.index, future_df['sales'], label='Future Predictions',␣
 ↪linestyle='--')
    plt.xlabel('Date')
    plt.ylabel('Sales')
    plt.title(f'Sales Forecast for Store {store_number}, Item {item_number}')
    plt.legend()
```

```
    plt.show()

    rmse = root_mean_squared_error(y, model.predict(X))
    print(f'RMSE: {rmse:.2f}')

    return prediction_df

# Example usage
store_number = 1
item_number = 1
predicted_sales = forecast_sales(maindataframe, store_number, item_number)
```

[ ]:
```
final_df = pd.DataFrame()

for store in range(1, 11):
    for item in range(1, 51):
        predicted_sales = forecast_sales(maindataframe, store, item)
        final_df = pd.concat([final_df, predicted_sales], axis=1)
```

[ ]:
```
final_df
```

[ ]:

	sales	sales	sales	sales	sales	sales	\
2018-01-01	14.787496	34.878716	19.566341	11.501266	14.153571	31.548494	
2018-01-02	15.092945	32.367382	23.920757	15.707705	14.929819	34.234055	
2018-01-03	13.475556	38.335934	22.843904	12.250526	13.282858	40.321430	
2018-01-04	14.362658	42.876656	22.691727	14.308525	15.019847	43.400272	
2018-01-05	20.010235	39.278217	26.595425	9.962386	15.903361	36.381271	
…	…	…	…	…	…	…	
2018-03-27	19.735460	45.198963	32.839222	19.640730	17.628244	45.054638	
2018-03-28	12.753129	44.997261	29.339094	19.490480	11.596881	34.729420	
2018-03-29	19.209019	47.158504	27.068308	20.225719	19.070772	54.649792	
2018-03-30	17.346434	54.385216	33.890732	16.832644	23.327826	68.294518	
2018-03-31	14.201283	59.506401	39.444145	17.284906	25.544809	57.285858	

	sales	sales	sales	sales	…	sales	\
2018-01-01	37.754684	33.635784	28.059975	46.527557	…	13.279940	
2018-01-02	36.567589	43.166641	29.528805	52.877590	…	9.496646	
2018-01-03	42.636143	52.416683	32.021828	48.935158	…	13.053600	
2018-01-04	40.354008	46.113312	36.995487	46.715424	…	21.378765	
2018-01-05	43.574196	50.726341	31.826172	48.764721	…	16.998423	
…	…	…	…	…	…	…	
2018-03-27	47.067238	64.118095	37.962463	67.336197	…	25.752546	
2018-03-28	49.129490	75.275131	39.117981	66.752396	…	20.757536	
2018-03-29	51.441830	73.916069	48.116421	66.296844	…	23.326527	
2018-03-30	53.553001	75.860695	47.955627	71.528458	…	26.932325	
2018-03-31	59.619549	83.738670	50.553822	70.413345	…	23.304806	

```
               sales      sales      sales       sales      sales      sales  \
2018-01-01  26.136559  38.747337  20.913124   55.835781  46.334557  15.373569
2018-01-02  30.190918  39.221722  24.909956   77.859802  44.497269  18.504930
2018-01-03  27.716278  40.790226  28.462656   67.701599  41.967964  13.409482
2018-01-04  24.435545  42.373150  26.935799   65.301750  48.928535  11.459268
2018-01-05  29.881964  39.301888  25.284472   63.245152  57.616417  22.129583
…                 …          …          …           …          …          …
2018-03-27  35.390202  48.892902  32.823410   85.420860  59.268944  24.949005
2018-03-28  30.849974  50.877602  40.546928   87.720772  54.378647  18.617756
2018-03-29  49.059872  59.140488  33.509060   82.930931  61.120949  23.353464
2018-03-30  47.671257  57.306313  39.499264  100.215012  66.277023  26.639299
2018-03-31  50.511719  51.595688  33.176395  111.326385  65.080338  23.772202

               sales      sales      sales
2018-01-01  35.035290  19.466610  41.544891
2018-01-02  45.030106  18.746435  52.668659
2018-01-03  35.973125  20.537277  65.627029
2018-01-04  38.143108  25.012955  65.780540
2018-01-05  42.918522  30.105364  70.824844
…                 …          …          …
2018-03-27  52.985157  30.736664  69.480331
2018-03-28  61.411270  34.841843  61.956146
2018-03-29  55.655064  39.309181  74.640793
2018-03-30  59.824955  43.338112  73.629013
2018-03-31  67.201660  29.367119  70.586700

[90 rows x 500 columns]
```

[ ]:
```
columns = []
for store in range(1, 11):
    for item in range(1, 51):
        columns.append((f'store_{store}', f'item_{item}'))

# Create MultiIndex and assign to DataFrame
multi_index_columns = pd.MultiIndex.from_tuples(columns, names=['store',␣
 ↪'item'])
final_df.columns = multi_index_columns

final_df.columns = ['_'.join(col) for col in final_df.columns]


print(final_df)
```

```
            store_1_item_1  store_1_item_2  store_1_item_3  store_1_item_4  \
2018-01-01       14.787496       34.878716       19.566341       11.501266
2018-01-02       15.092945       32.367382       23.920757       15.707705
2018-01-03       13.475556       38.335934       22.843904       12.250526
```

```
2018-01-04       14.362658       42.876656       22.691727       14.308525
2018-01-05       20.010235       39.278217       26.595425        9.962386
…                      …               …               …              …
2018-03-27       19.735460       45.198963       32.839222       19.640730
2018-03-28       12.753129       44.997261       29.339094       19.490480
2018-03-29       19.209019       47.158504       27.068308       20.225719
2018-03-30       17.346434       54.385216       33.890732       16.832644
2018-03-31       14.201283       59.506401       39.444145       17.284906

            store_1_item_5  store_1_item_6  store_1_item_7  store_1_item_8  \
2018-01-01       14.153571       31.548494       37.754684       33.635784
2018-01-02       14.929819       34.234055       36.567589       43.166641
2018-01-03       13.282858       40.321430       42.636143       52.416683
2018-01-04       15.019847       43.400272       40.354008       46.113312
2018-01-05       15.903361       36.381271       43.574196       50.726341
…                      …               …               …              …
2018-03-27       17.628244       45.054638       47.067238       64.118095
2018-03-28       11.596881       34.729420       49.129490       75.275131
2018-03-29       19.070772       54.649792       51.441830       73.916069
2018-03-30       23.327826       68.294518       53.553001       75.860695
2018-03-31       25.544809       57.285858       59.619549       83.738670

            store_1_item_9  store_1_item_10  …  store_10_item_41  \
2018-01-01       28.059975        46.527557  …         13.279940
2018-01-02       29.528805        52.877590  …          9.496646
2018-01-03       32.021828        48.935158  …         13.053600
2018-01-04       36.995487        46.715424  …         21.378765
2018-01-05       31.826172        48.764721  …         16.998423
…                      …                …  …               …
2018-03-27       37.962463        67.336197  …         25.752546
2018-03-28       39.117981        66.752396  …         20.757536
2018-03-29       48.116421        66.296844  …         23.326527
2018-03-30       47.955627        71.528458  …         26.932325
2018-03-31       50.553822        70.413345  …         23.304806

            store_10_item_42  store_10_item_43  store_10_item_44  \
2018-01-01         26.136559         38.747337         20.913124
2018-01-02         30.190918         39.221722         24.909956
2018-01-03         27.716278         40.790226         28.462656
2018-01-04         24.435545         42.373150         26.935799
2018-01-05         29.881964         39.301888         25.284472
…                        …                 …                 …
2018-03-27         35.390202         48.892902         32.823410
2018-03-28         30.849974         50.877602         40.546928
2018-03-29         49.059872         59.140488         33.509060
2018-03-30         47.671257         57.306313         39.499264
2018-03-31         50.511719         51.595688         33.176395
```

```
            store_10_item_45  store_10_item_46  store_10_item_47  \
2018-01-01         55.835781         46.334557         15.373569
2018-01-02         77.859802         44.497269         18.504930
2018-01-03         67.701599         41.967964         13.409482
2018-01-04         65.301750         48.928535         11.459268
2018-01-05         63.245152         57.616417         22.129583
…                        …                 …                 …
2018-03-27         85.420860         59.268944         24.949005
2018-03-28         87.720772         54.378647         18.617756
2018-03-29         82.930931         61.120949         23.353464
2018-03-30        100.215012         66.277023         26.639299
2018-03-31        111.326385         65.080338         23.772202

            store_10_item_48  store_10_item_49  store_10_item_50
2018-01-01         35.035290         19.466610         41.544891
2018-01-02         45.030106         18.746435         52.668659
2018-01-03         35.973125         20.537277         65.627029
2018-01-04         38.143108         25.012955         65.780540
2018-01-05         42.918522         30.105364         70.824844
…                        …                 …                 …
2018-03-27         52.985157         30.736664         69.480331
2018-03-28         61.411270         34.841843         61.956146
2018-03-29         55.655064         39.309181         74.640793
2018-03-30         59.824955         43.338112         73.629013
2018-03-31         67.201660         29.367119         70.586700

[90 rows x 500 columns]
```

[ ]:
```
final_df.to_csv('finalresult.csv')
```

[ ]:
```
required = pd.read_csv('demand-forecasting-kernels-only/test.csv')
```

[ ]:
```
# Ensure 'date' column in 'required' is in datetime format
required['date'] = pd.to_datetime(required['date'])

# Create the 'ans' column in 'required'
required['ans'] = required.apply(lambda row: final_df.loc[row['date'],␣
 ↪f'store_{row["store"]}_item_{row["item"]}'], axis=1)
```

[ ]:
```
required
```

[ ]:
```
          id       date  store  item        ans
0          0 2018-01-01      1     1  14.787496
1          1 2018-01-02      1     1  15.092945
2          2 2018-01-03      1     1  13.475556
3          3 2018-01-04      1     1  14.362658
4          4 2018-01-05      1     1  20.010235
…        …          …     …    …         …
```

```
44995  44995 2018-03-27     10    50  69.480331
44996  44996 2018-03-28     10    50  61.956146
44997  44997 2018-03-29     10    50  74.640793
44998  44998 2018-03-30     10    50  73.629013
44999  44999 2018-03-31     10    50  70.586700

[45000 rows x 5 columns]
```

[ ]:
```
required = required.drop(columns='date')
```

[ ]:
```
required = required.drop(columns='store')
```

[ ]:
```
required = required.drop(columns='item')
```

[ ]:
```
required['sales'] = required['ans']
required = required.drop(columns='ans')
```

[ ]:
```
required
```

[ ]:
```
           id      sales
0           0  14.787496
1           1  15.092945
2           2  13.475556
3           3  14.362658
4           4  20.010235
…         …         …
44995   44995  69.480331
44996   44996  61.956146
44997   44997  74.640793
44998   44998  73.629013
44999   44999  70.586700

[45000 rows x 2 columns]
```

[ ]:
```
required.index.name = 'id'
```

[ ]:
```
required = required.drop(columns='id')
```

[ ]:
```
required
```

[ ]:
```
            sales
id
0       14.787496
1       15.092945
2       13.475556
3       14.362658
4       20.010235
…           …
```

```
44995  69.480331
44996  61.956146
44997  74.640793
44998  73.629013
44999  70.586700

[45000 rows x 1 columns]
```

[ ]:
```
required.to_csv('submission.csv')
```

[ ]:
```

```