05-01-Compare-models

April 12, 2024

0.0.1 Compare models

[37]:
```
import pandas as pd
import numpy as np
import warnings
import tensorflow as tf
import tensorboard
```

0.0.2 Load the data

[38]:
```
### Load stratified data
strat_splits = []
for i in range(10):
    split = []
    for j in range(2):
        split.append(pd.read_pickle(f'pickled-data/df_{i}-{j}.pkl'))
    strat_splits.append(split)
```

[75]:
```
strat_train_set, strat_test_set = strat_splits[0] # train with 100000 for␣
  ↪reasonable amount of training
```

[74]:
```
# validation data
trips_test = strat_test_set.drop(columns=['trip_duration'])          #␣
  ↪predictors
trips_test_label = strat_test_set["trip_duration"]                   # targets
trips_test_label = trips_test_label/pd.Timedelta(minutes=1)
```

[145]:
```
strat_new_train, strat_new_test = strat_splits[5] # Model has never seen this␣
  ↪data for prediction
new_test = strat_new_test.drop(columns=['trip_duration'])          # predictors
new_test_label = strat_new_test[["trip_duration"]]               # targets
new_test_label = new_test_label/pd.Timedelta(minutes=1)
```

Process features

[41]:
```
def extract_features(trips):
    trips['pickup_weekday'] = trips['tpep_pickup_datetime'].dt.weekday
    trips['pickup_hour'] = trips['tpep_pickup_datetime'].dt.hour
    trips['pickup_minute'] = trips['tpep_pickup_datetime'].dt.minute
    return trips
```

[42]:

```
# a utility function to drop features
def feature_selection(dataframe, attributes=[]):
    return dataframe.drop(columns=attributes)
```

[43]:

```
def type_casting(dataframe, attribute, type):
    dataframe[f"{attribute}"] = dataframe[[f"{attribute}"]].astype(f"{type}")
    return dataframe
```

[87]:

```
# Extract features from datetime columns of pickup
trips_test = extract_features(trips_test)
new_test = extract_features(new_test)
```

[88]:

```
# drop the tpep_pickup_datetime columns and date columns (used for joining)
drop_dates = ["tpep_pickup_datetime", "date"]
trips_test = feature_selection(trips_test, drop_dates)
new_test = feature_selection(new_test, drop_dates)

# drop irrelevant data columns
irrelevant_attr = ["payment_type", "VendorID", "RatecodeID"]
trips_test = feature_selection(trips_test, irrelevant_attr)
new_test = feature_selection(new_test, irrelevant_attr)

# drop columns with significant missing values i.e., almost equal to the␣
 ↪dataset size
significant_nulls = ["wpgt", "snow", "prcp", "tsun", "wdir", "airport_fee"]
trips_test = feature_selection(trips_test, significant_nulls)
new_test = feature_selection(new_test, significant_nulls)
```

[89]:

```
# cast dates to a numeral
trips_test = type_casting(trips_test, "tpep_dropoff_datetime", "int64")
new_test = type_casting(new_test, "tpep_dropoff_datetime", "int64")
```

[90]:

```
trips_test.head(2)
```

[90]:

```
         tpep_dropoff_datetime  passenger_count  trip_distance  \
4827251       1579809623000000              1.0            5.5
3890488       1579392861000000              1.0            0.7

        store_and_fwd_flag  PULocationID  DOLocationID  fare_amount  extra  \
4827251                  N           234            24         22.0    3.5
3890488                  N           230           164          4.0    0.5

         mta_tax  tip_amount  …  total_amount  congestion_surcharge  tavg  \
4827251      0.5        5.25  …         31.55                   2.5   3.5
3890488      0.5        1.95  …          9.75                   2.5   4.0
```

```
         tmin  tmax  wspd    pres  pickup_weekday  pickup_hour  pickup_minute
4827251   0.0   7.2   7.5  1029.4               3           19             32
3890488   0.6   7.2  10.2  1008.9               6            0             12

[2 rows x 22 columns]
```

[92]:

```
new_test.head(2)
```

[92]:

```
         tpep_dropoff_datetime  passenger_count  trip_distance  \
1906266       1580421173000000              2.0           2.30
6073032       1580230252000000              2.0           1.52

        store_and_fwd_flag  PULocationID  DOLocationID  fare_amount  extra  \
1906266                  N           144           164         11.5    3.0
6073032                  N           237           236          7.5    1.0

         mta_tax  tip_amount  …  congestion_surcharge  tavg  tmin  tmax  \
1906266      0.5        0.00  …                   2.5   1.3  -1.7   4.4
6073032      0.5        2.36  …                   2.5   4.9   3.9   7.2

         wspd    pres  pickup_time_cat  pickup_weekday  pickup_hour  \
1906266  10.7  1026.3          evening               3           21
6073032   6.3  1010.2        afternoon               1           16

        pickup_minute
1906266            38
6073032            43

[2 rows x 23 columns]
```

Standardize the input data

[48]:

```
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

[49]:

```
# numerical transformer
num_attributes = list(trips_test.select_dtypes(np.number).columns)
num_pipeline = make_pipeline(SimpleImputer(strategy="mean"),
                             StandardScaler())

# categorical transformer
cat_attributes = ['store_and_fwd_flag']
cat_pipeline = make_pipeline(SimpleImputer(strategy="most_frequent"),
                                      OneHotEncoder(handle_unknown="ignore"))
```

[50]:

```
# combined Transformation pipelines
preprocessing = ColumnTransformer([
        ("num", num_pipeline, num_attributes),
        ("cat", cat_pipeline, cat_attributes),
    ])
```

[51]:

```
trips_test_prepared = preprocessing.fit_transform(trips_test)
df_trips_test_prepared = pd.DataFrame(trips_test_prepared,
                                       columns=preprocessing.
 ↪get_feature_names_out(),
                                       index=trips_test.index)
df_trips_test_prepared.head(2)
```

[51]:

```
         num__tpep_dropoff_datetime  num__passenger_count  num__trip_distance  \
4827251                    0.753343             -0.449526            0.683024
3890488                    0.207758             -0.449526           -0.573670

         num__PULocationID  num__DOLocationID  num__fare_amount  num__extra  \
4827251           1.055993          -1.983284          0.773485    1.895701
3890488           0.995023           0.017897         -0.721763   -0.488914

         num__mta_tax  num__tip_amount  num__tolls_amount  …  num__tmin  \
4827251       0.10568         1.129379          -0.179408  …  -0.406976
3890488       0.10568        -0.088479          -0.179408  …  -0.234142

         num__tmax  num__wspd  num__pres  num__pickup_weekday  \
4827251  -0.078756  -0.904512   0.936669             0.005780
3890488  -0.078756  -0.119922  -1.388218             1.628044

         num__pickup_hour  num__pickup_minute  cat__store_and_fwd_flag_N  \
4827251          0.868344            0.136937                        1.0
3890488         -2.361503           -1.016936                        1.0

         cat__store_and_fwd_flag_Y  cat__store_and_fwd_flag_None
4827251                        0.0                           0.0
3890488                        0.0                           0.0

[2 rows x 24 columns]
```

[93]:

```
new_test_prepared = preprocessing.fit_transform(new_test)
df_new_test_prepared = pd.DataFrame(new_test_prepared,
                                       columns=preprocessing.
 ↪get_feature_names_out(),
                                       index=new_test.index)
df_new_test_prepared.head(2)
```

```
[93]:          num__tpep_dropoff_datetime  num__passenger_count  num__trip_distance  \
1906266                    1.556731              0.422239           -0.156173
6073032                    1.306750              0.422239           -0.358961

         num__PULocationID  num__DOLocationID  num__fare_amount  num__extra  \
1906266          -0.316725           0.018986         -0.100251    1.492665
6073032           1.102748           1.048328         -0.433631   -0.091544

         num__mta_tax  num__tip_amount  num__tolls_amount  …  num__tmin  \
1906266      0.105465        -0.800881          -0.215514  …  -0.896012
6073032      0.105465         0.061439          -0.215514  …   0.716600

         num__tmax  num__wspd  num__pres  num__pickup_weekday  \
1906266  -0.908246   0.023658   0.584941             0.005604
6073032  -0.078577  -1.252864  -1.240534            -1.074916

         num__pickup_hour  num__pickup_minute  cat__store_and_fwd_flag_N  \
1906266          1.208005            0.482940                        1.0
6073032          0.358228            0.771112                        1.0

         cat__store_and_fwd_flag_Y  cat__store_and_fwd_flag_None
1906266                        0.0                           0.0
6073032                        0.0                           0.0

[2 rows x 24 columns]
```

0.0.3 load the models and evaluate

```
[52]: import os
from tensorflow.keras.models import load_model

def evaluate_models(models_directory, test_data, test_labels):
    model_files = os.listdir(models_directory)
    evaluation_results = []

    for model_file in model_files:
        model_path = os.path.join(models_directory, model_file)
        try:
            model = load_model(model_path)
            loss, accuracy = model.evaluate(test_data, test_labels, verbose=0)
            evaluation_results.append({"Model": model_file, "Loss": loss,␣
 ↪"Accuracy": accuracy})
        except Exception as e:
            print(f"Error loading or evaluating model {model_file}: {e}")

    # Create a DataFrame from the evaluation results
    df_results = pd.DataFrame(evaluation_results)
    return df_results
```

[53]:
```
eval_results = evaluate_models("models/", df_trips_test_prepared,␣
 ↪trips_test_label)
eval_results
```

[53]:
```
                                               Model         Loss   Accuracy
0                          04-02-MLP-ADAM-MAE.keras     3.922165  62.734173
1                          04-02-MLP-ADAM-MSE.keras  3939.668457  62.766777
2                       04-02-MLP-RMSProp-MAE.keras     3.976194  62.772575
3                       04-02-MLP-RMSProp-MSE.keras  3939.199951  62.763046
4                           04-02-MLP-SGD-MAE.keras     4.049551  62.782619
5                           04-02-MLP-SGD-MSE.keras  3991.377197  63.177349
6      04-03-MLP-with-no-hidden-layers-ADAM-MAE.keras     5.389725  63.646450
7      04-03-MLP-with-no-hidden-layers-ADAM-MSE.keras  3947.220703  62.826912
8   04-03-MLP-with-no-hidden-layers-RMSProp-MAE.keras     5.392833  63.655872
9   04-03-MLP-with-no-hidden-layers-RMSProp-MSE.keras  3952.218262  62.866669
10      04-03-MLP-with-no-hidden-layers-SGD-MAE.keras  3947.801514  62.831532
11      04-03-MLP-with-no-hidden-layers-SGD-MSE.keras  3947.801270  62.831532
12                         04-04-DNN-ADAM-MAE.keras     4.912194  62.938251
13                         04-04-DNN-ADAM-MSE.keras  3945.618408  62.814159
14                      04-04-DNN-RMSProp-MAE.keras     4.698244  62.870094
15                      04-04-DNN-RMSProp-MSE.keras  3957.509766  62.908741
16                          04-04-DNN-SGD-MAE.keras     5.425389  62.974220
17                          04-04-DNN-SGD-MSE.keras  3937.330566  62.748154
```

From above evaluation the best model is

[67]:
```
best_model = eval_results.loc[eval_results['Accuracy'] ==␣
 ↪eval_results["Accuracy"].max()]
best_model
```

[67]:
```
                                               Model      Loss   Accuracy
8  04-03-MLP-with-no-hidden-layers-RMSProp-MAE.keras  5.392833  63.655872
```

[70]:
```
best_model.Model.iloc[0]
```

[70]:
```
'04-03-MLP-with-no-hidden-layers-RMSProp-MAE.keras'
```

[71]:
```
model = load_model(f"models/{best_model.Model.iloc[0]}")
```

[119]:
```
trips_new = df_new_test_prepared[:100] # predict the first 5
trips_pred = model.predict(trips_new)
```

```
4/4               0s 7ms/step
```

[120]:
```
df_predicted = pd.DataFrame(trips_pred, columns=["predicted"],␣
 ↪index=new_test_label[:100].index)
```

[154]:

```
df_predicted.head(5)
```

[154]:

```
         predicted
1906266  13.806322
6073032   8.283689
2031201   5.741024
1684989   6.760906
4968211   9.686960
```

[155]:

```
new_test_label["predicted"] = df_predicted[["predicted"]]
new_test_label.head(5)
```

[155]:

```
         trip_duration  predicted
1906266      14.350000  13.806322
6073032       7.683333   8.283689
2031201       5.433333   5.741024
1684989       6.666667   6.760906
4968211       9.966667   9.686960
```

[132]:

```
temp = new_test_label
temp["predicted"] = df_predicted["predicted"]
```