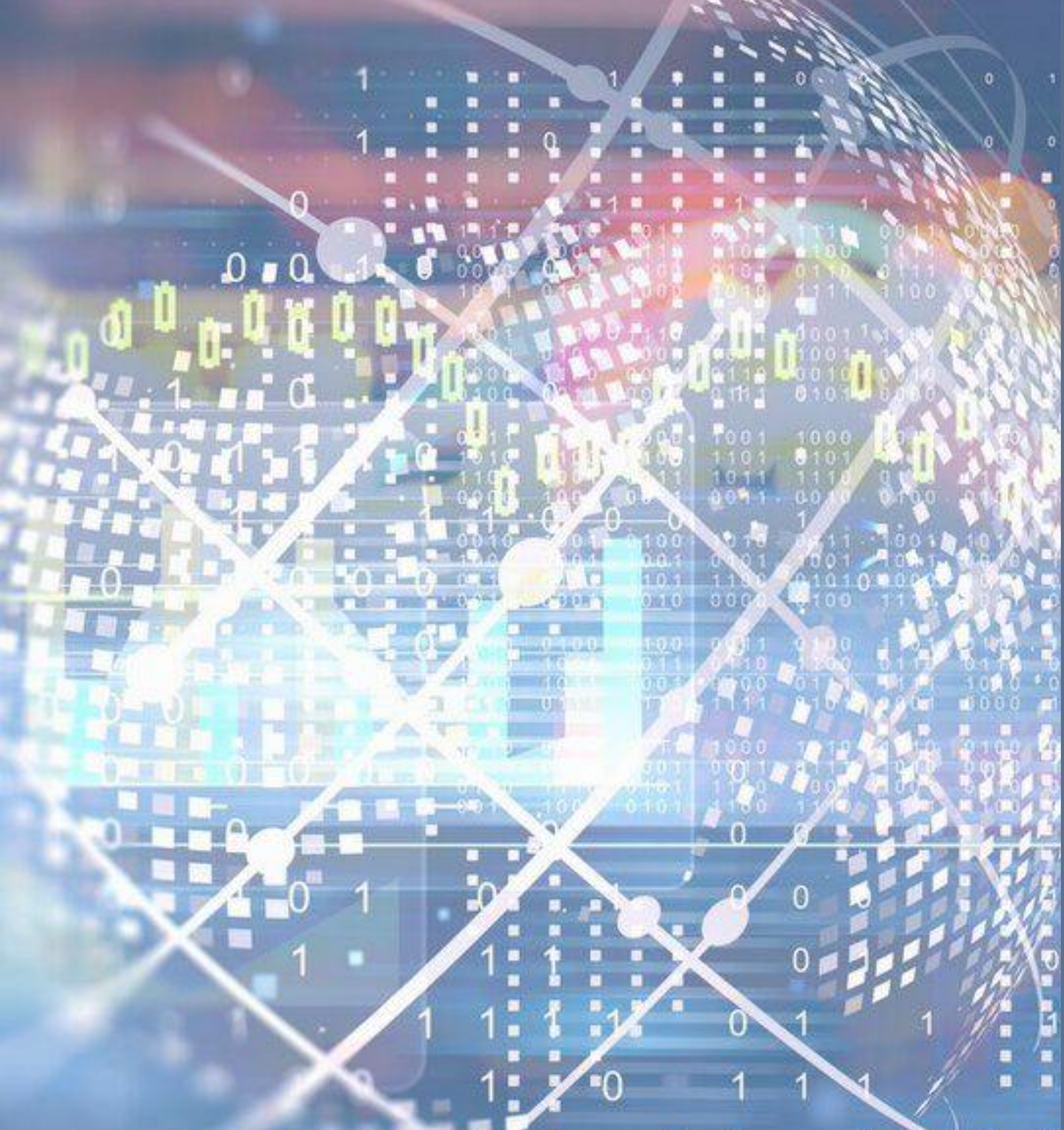

Graded Activity - Case Study 6.1

Using Feature Engineering to Predict NYC Taxi Trips

Instructions

1. Save this template on your computer and change the name of the file as follows: YourLastName_FirstName_CS_6

*Note: You will not be able to retrieve this document from the edX platform after you submit it. Please, save this document in a central location for future reference.

2. This is an individual activity. A scoring rubric can be downloaded from the CS6.1 section on the wiki. This rubric will help you to successfully complete your activity and grade others.

3. Read the CS self-help documentation and follow the steps and answer the questions on this activity sheet.

3. If you have any questions, feel free to ask the TAs in the forum space created for this activity.

4. Upload your document as a .pdf or as a .ppt / .pptx

*Note: edX has a **10MB** file size limit for document submission. If you have used large image(s), you may need to resize before submitting, OR you may simply include a web URL for the image in the image location on this activity sheet. Be sure to submit your assignment at least one hour before the deadline to provide time for troubleshooting.

5. The edX system will automatically assign you a Case Study from one of your peers. Follow the rubric and assess your peer.

Case Study 6.1

- **1.1 Paste in the first 5 Cutoff Times**

Code

```
cutoff_time = trips[['id', 'pickup_datetime']]
cutoff_time = cutoff_time[cutoff_time['pickup_datetime'] > "2016-01-12"]
```

Output (the first 5 Cutoff Times)

```
   id      pickup_datetime
56311 2016-01-12 00:00:25
56312 2016-01-12 00:02:09
56313 2016-01-12 00:02:25
56314 2016-01-12 00:02:41
56315 2016-01-12 00:03:44
```

Comments:

- The data contains trips with pickup_datetimne from 2016/01/01 to 2016/06/30
- Cutoff Time criteria: only select trips that started after January 12th, 2016

- **1.2 Do these cutoff times make sense? Why did we choose these for the cutoff times?**

Cutoff times make sense. They are needed to make the duration prediction using data before the trip starts; that is why pickup_datetime timestamp is used as cutoff times.

Case Study 6.1

- **2.1 What was the Modeling Score after your last training round?**

Not including transform primitives or aggregate primitives

Code

```
features = ft.dfs(entities=entities,
                  relationships=relationships,
                  target_entity="trips",
                  trans_primitives=[],
                  agg_primitives=[],
                  ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                              "dropoff_latitude", "dropoff_longitude"]},
                  features_only=True)
```

```
feature_matrix = compute_features(features, cutoff_time)
```

```
# separates the whole feature matrix into train data feature matrix,
# train data labels, and test data feature matrix
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train+1)
y_test = np.log(y_test+1)
```

```
model = GradientBoostingRegressor(verbose=True)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Case Study 6.1

- 2.1 What was the Modeling Score after your last training round? (cont.)
  Not including transform primitives or aggregate primitives

Output

```
Number of features: 11

[<Feature: vendor_id>,
 <Feature: passenger_count>,
 <Feature: dropoff_neighborhood>,
 <Feature: payment_type>,
 <Feature: pickup_neighborhood>,
 <Feature: trip_duration>,
 <Feature: trip_distance>,
 <Feature: dropoff_neighborhoods.longitude>,
 <Feature: pickup_neighborhoods.latitude>,
 <Feature: dropoff_neighborhoods.latitude>,
 <Feature: pickup_neighborhoods.longitude>]
```

Iter	Train Loss	Remaining Time
1	0.4925	2.74m
2	0.4333	2.73m
3	0.3843	2.68m
4	0.3446	2.66m
5	0.3119	2.62m
6	0.2852	2.59m
7	0.2634	2.56m
8	0.2454	2.53m
9	0.2305	2.51m
10	0.2183	2.47m
20	0.1676	2.21m
30	0.1580	1.91m
40	0.1543	1.58m
50	0.1523	1.27m
60	0.1507	59.41s
70	0.1497	43.83s
80	0.1489	29.28s
90	0.1481	14.45s
100	0.1475	0.00s

Modeling Score: 0.71341469457415285

- 2.2 Hypothesize on how including more robust features will change the accuracy.

Increasing more robust features, for example, transform or aggregate features, probably increases the modeling score or decreases RMSE.

Case Study 6.1

- 3.1 Paste in 5 of the generated features
- 3.2 What are these features called and how are they generated?

Creating **transform features** using **transform primitives**.
A transform primitive called "**weekend**" can be applied to any `datetime` column in the data. In this specific data, there are two datetime columns `pickup_datetime` and `dropoff_datetime`. The tool automatically creates features using the primitive.

Code

```
trans_primitives = [Weekend]
```

```
features = ft.dfs(entities=entities,
                  relationships=relationships,
                  target_entity="trips",
                  trans_primitives=trans_primitives,
                  agg_primitives=[],
                  ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                              "dropoff_latitude", "dropoff_longitude"]},
                  features_only=True)
```

```
feature_matrix = compute_features(features, cutoff_time)
```

```
# separates the whole feature matrix into train data feature matrix,
# train data labels, and test data feature matrix
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train+1)
y_test = np.log(y_test+1)
```

```
model = GradientBoostingRegressor(verbose=True)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Output

```
Number of features: 13

[<Feature: vendor_id>,
 <Feature: passenger_count>,
 <Feature: dropoff_neighborhood>,
 <Feature: payment_type>,
 <Feature: pickup_neighborhood>,
 <Feature: trip_duration>,
 <Feature: trip_distance>,
 <Feature: dropoff_neighborhoods.longitude>,
 <Feature: dropoff_neighborhoods.latitude>,
 <Feature: pickup_neighborhoods.longitude>,
 <Feature: pickup_neighborhoods.latitude>,
 <Feature: IS_WEEKEND(pickup_datetime)>,
 <Feature: IS_WEEKEND(dropoff_datetime)>]
```

Modeling Score: `0.7220107526801755 6`

Case Study 6.1

Adding transform primitives called “**Minute, Hour, Day, Week, Month, Weekday, Weekend**” to the two datetime columns pickup_datetime and dropoff_datetime.

Code

```
trans_primitives = [Minute, Hour, Day, Week, Month, Weekday, Weekend]

features = ft.dfs(entities=entities,
                  relationships=relationships,
                  target_entity="trips",
                  trans_primitives=trans_primitives,
                  agg_primitives=[],
                  ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                              "dropoff_latitude", "dropoff_longitude"]},
                  features_only=True)

feature_matrix = compute_features(features, cutoff_time)

# separates the whole feature matrix into train data feature matrix,
# train data labels, and test data feature matrix
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train+1)
y_test = np.log(y_test+1)

model = GradientBoostingRegressor(verbose=True)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Output

```
Number of features: 25

[<Feature: passenger_count>,
 <Feature: dropoff_neighborhood>,
 <Feature: payment_type>,
 <Feature: vendor_id>,
 <Feature: pickup_neighborhood>,
 <Feature: trip_duration>,
 <Feature: trip_distance>,
 <Feature: DAY(pickup_datetime)>,
 <Feature: dropoff_neighborhoods.latitude>,
 <Feature: WEEK(dropoff_datetime)>,
 <Feature: HOUR(pickup_datetime)>,
 <Feature: WEEKDAY(dropoff_datetime)>,
 <Feature: WEEKDAY(pickup_datetime)>,
 <Feature: MONTH(pickup_datetime)>,
 <Feature: WEEK(pickup_datetime)>,
 <Feature: DAY(dropoff_datetime)>,
 <Feature: MONTH(dropoff_datetime)>,
 <Feature: pickup_neighborhoods.latitude>,
 <Feature: HOUR(dropoff_datetime)>,
 <Feature: pickup_neighborhoods.longitude>,
 <Feature: IS_WEEKEND(pickup_datetime)>,
 <Feature: MINUTE(pickup_datetime)>,
 <Feature: MINUTE(dropoff_datetime)>,
 <Feature: dropoff_neighborhoods.longitude>,
 <Feature: IS_WEEKEND(dropoff_datetime)>]
```

Modeling Score: 0.7755608981558122

Case Study 6.1

- **4.1 What was the Modeling Score after your last training round when including the transform primitives?**

Modeling Score (without transform features): 0.71341469457415285

Modeling Score (with transform feature "**Weekend**"): 0.72201075268017556

Modeling Score (with transform features "**Minute, Hour, Week, Month, Weekday, Weekend**"): 0.7755608981558122

- **4.2 Comment on how the modeling accuracy differs when including the Transform features.**

Adding transform features increases the modeling score/accuracy or decreases RMSE

Case Study 6.1

- **5.1 What was the Modeling Score after your last training round when including the aggregate transforms?**

Including both transform primitives and aggregate primitives

Code

```
trans_primitives = [Minute, Hour, Day, Week, Month, Weekday, Weekend]
aggregation_primitives = [Count, Sum, Mean, Median, Std, Max, Min]

features = ft.dfs(entities=entities,
                  relationships=relationships,
                  target_entity="trips",
                  trans_primitives=trans_primitives,
                  agg_primitives=aggregation_primitives,
                  ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                              "dropoff_latitude", "dropoff_longitude"]},
                  features_only=True)
```

transformative primitives

aggregative primitives

```
feature_matrix = compute_features(features, cutoff_time)
```

```
# separates the whole feature matrix into train data feature matrix,
# train data labels, and test data feature matrix
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train+1)
y_test = np.log(y_test+1)
```

```
model = GradientBoostingRegressor(verbose=True)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Modeling Score: 0.7780888563898487

Increase not significantly w.r.t. the previous run with transformative primitives

Case Study 6.1

- 5.2 How do these aggregate transforms impact performance? How do they impact training time?

Not including transform primitives or aggregate primitives

Iter	Train Loss	Remaining Time
1	0.4925	2.74m
2	0.4333	2.73m
3	0.3843	2.68m
4	0.3446	2.66m
5	0.3119	2.62m
6	0.2852	2.59m
7	0.2634	2.56m
8	0.2454	2.53m
9	0.2305	2.51m
10	0.2183	2.47m
20	0.1676	2.21m
30	0.1580	1.91m
40	0.1543	1.58m
50	0.1523	1.27m
60	0.1507	59.41s
70	0.1497	43.83s
80	0.1489	29.28s
90	0.1481	14.45s
100	0.1475	0.00s

Modeling Score: 0.7134146945741528 5

Including transform primitives

Iter	Train Loss	Remaining Time
1	0.4925	4.03m
2	0.4333	3.96m
3	0.3843	3.86m
4	0.3444	3.85m
5	0.3117	3.85m
6	0.2848	3.79m
7	0.2620	3.73m
8	0.2435	3.76m
9	0.2282	3.76m
10	0.2152	3.69m
20	0.1588	3.16m
30	0.1415	2.64m
40	0.1332	2.21m
50	0.1283	1.78m
60	0.1252	1.38m
70	0.1227	1.01m
80	0.1207	39.98s
90	0.1191	19.72s
100	0.1177	0.00s

Modeling Score: 0.7755608981558122

Including both transform primitives and aggregate primitives

Iter	Train Loss	Remaining Time
1	0.4925	10.14m
2	0.4333	10.11m
3	0.3843	10.35m
4	0.3444	10.26m
5	0.3117	10.23m
6	0.2848	10.20m
7	0.2620	10.07m
8	0.2435	10.00m
9	0.2282	9.95m
10	0.2152	9.85m
20	0.1585	9.08m
30	0.1420	7.79m
40	0.1332	6.50m
50	0.1271	5.26m
60	0.1238	4.13m
70	0.1211	3.06m
80	0.1191	2.02m
90	0.1176	1.00m
100	0.1163	0.00s

Modeling Score: 0.7780888563898487

Comments:

- Aggregate primitives increase training time significantly. Training time increases from 2.74 mins with out transform or aggregate primitives to 4.03 mins with transform primitives, and to 10.14 mins with both transform and aggregate primitives.
- Modeling score increases or RMSE decreases significantly by including transform and aggregate primitives. However, adding aggregate primitives into the model which already includes transform primitive improves modeling score or decreases RMSE insignificantly (model score increases from 0.77556 to 0.77808)