Congratulations! You passed!

Next Item

1 / 1 point

1. You are training a model to predict how long it will take to sell a house. The list price of the house, with numeric $20,000 to $500,000 values, is one of the inputs to the model. Which of these is a good practice?

- [x] Rescale the real valued feature like a price to a range from 0 to 1

Correct

- [ ] Rescale the real valued feature like a price to a range from 0 to $100,000
- [ ] Rescale the real valued feature like a price to a categorical range from low, medium, high

1 / 1 point

2. Which of these tools are commonly used for data pre-processing? (Select 3 correct responses)

- [ ] Bigtable

Un-selected is correct

- [x] Apache Beam

Correct

- [x] BigQuery

Correct

- [ ] Google Cloud Storage

Un-selected is correct

- [x] TensorFlow

Correct

1 / 1 point

3. Which one of these is NOT something you would commonly do in data preprocessing?

- [ ] Compute vocabularies for categorical columns
- [ ] Compute aggregate statistics for numeric columns
- [ ] Compute time-windowed statistics (e.g. number of products sold in previous hour) for use as input features
- [x] Tune your ML model hyperparameters

Correct
Correct - this will come later

- [ ] Remove examples that you don't want to train on

1 / 1 point

4. In your TensorFlow model you are calculating the distance between two points on a map as a new feature. How do you ensure the preprocessing you're doing for model training is also do the exact same way in prediction?

1. Example of preprocessing in TensorFlow input_fn

```
def add_engineered(features):
    lat1 = features['pickuplat']
    ...
    dist = tf.sqrt(latdiff*latdiff + londiff*londiff)
    features['euclidean'] = dist
```

```
    return features
```

- Wrap features in training/evaluation input function:

```
def input_fn():
  features = ...
  label = ...
  return add_engineered(features), label
```

- Wrap features in serving input function:

```
def serving_input_fn():
    feature_placeholders = ...
    features = ...
    return tf.estimator.export.ServingInputReceiver(
      add_engineered(features), feature_placeholders)
```

- Wrap features in training/evaluation input function AND wrap features in serving input function:

```
def input_fn():
  features = ...
  label = ...
  return add_engineered(features), label
```

```
def serving_input_fn():
    feature_placeholders = ...
    features = ...
    return tf.estimator.export.ServingInputReceiver(
      add_engineered(features), feature_placeholders)
```

Correct

✔ 1 / 1 point

5. The below code preprocesses the latitude and longitude using feature columns. What is the point of the 38.0 and 42.0 in the column buckets?

```
def build_estimator(model_dir, nbuckets):
  latbuckets = np.linspace(38.0, 42.0, nbuckets).tolist()
  b_plat = tf.feature_column.bucketized_column(plat, latbuckets)
  b_dlat = tf.feature_column.bucketized_column(dlat, latbuckets)

  return tf.estimator.LinearRegressor(
      model_dir=model_dir,
      feature_columns=[..., b_plat, b_dlat, ...])
```

- These define how many samples to put into each bucket (at least 38 but no more than 42 in each small bucket)
- These parameters ensure all latitudes in the raw dataset do not include 38 and 42 which we want to exclude for this dataset.
- Latitudes must be between 38 and 42 will be discretized into the specified number of bins.

Correct

✔ 1 / 1 point

6. What are two advantages of using TensorFlow to preprocess your code instead of building an Apache Beam pipeline? (Select two correct responses)

- [x] In TensorFlow the same pipelines can be used in both training and serving

Correct
This is because the TensorFlow function is part of the model graph

- [x] In TensorFlow you will have access to helper APIs to help automatically bucketize and process features instead of writing your own java or python code

Correct

- [ ] In TensorFlow the Apache Beam pipeline code is automatically generated for you

Un-selected is correct

---

1 / 1 point

7. What is one key advantage of preprocessing your features using Apache Beam?

- Apache Beam code is often harder to maintain and run at scale than BigQuery preprocessing pipelines
- The same code you use to preprocess features in training and evaluation can also be used in serving

Correct

- Apache Beam transformations are written in Standard SQL which is scalable and easy to author

---