```
import tensorflow as tf
print(tf.__version__)
```

```
2.8.0
```

```
h = "hellow"
w = "world"

hw = h+w
print(hw)
```

```
hellowworld
```

```
 h = tf.constant("hello")
 w = tf.constant("world")

 hw = h+w
 print(hw)
```

```
tf.Tensor(b'helloworld', shape=(), dtype=string)
```

```
with tf.compat.v1.Session() as sess:
  h1 = tf.constant("hello")
  w2 = tf.constant("world")
  hw = h1+w2
  ans = sess.run(hw)

print(ans)
```

```
b'helloworld'
```

```
with tf.compat.v1.Session() as sess:
  a = tf.constant(4)
  b = tf.constant(2)
  c = a*b
  print(sess.run(c))
```

```
8
```

```
data = tf.compat.v1.random_normal((1,5),0,1)
data
```

```
<tf.Tensor: shape=(1, 5), dtype=float32, numpy=
array([[-0.41924727, -0.03333364, -0.25328106, -3.2080953 , -0.3210965 ]],
      dtype=float32)>
```

```
var = tf.Variable(data,name = "var")
var
```

```
<tf.Variable 'var:0' shape=(1, 5) dtype=float32, numpy=
array([[-0.41924727, -0.03333364, -0.25328106, -3.2080953 , -0.3210965 ]],
      dtype=float32)>
```

```
string = tf.Variable("this is tensor",tf.string)
tf.rank(string)
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=0>
```

```
r = tf.Variable([['a'],['b']],tf.string)
```

```
r.shape
```

```
TensorShape([2, 1])
```

```
string.shape
```

```
TensorShape([])
```

```
tensor1 = tf.ones([1,2,3])
```

```
print(tensor1)
```

```
tf.Tensor(
[[[1. 1. 1.]
  [1. 1. 1.]]], shape=(1, 2, 3), dtype=float32)
```

```
tensor2 = tf.reshape(tensor1,[2,3,1])
print(tensor2)
```

```
tf.Tensor(
[[[1.]
  [1.]
  [1.]]

 [[1.]
  [1.]
  [1.]]], shape=(2, 3, 1), dtype=float32)
```

```
tensor3 = tf.reshape(tensor1,[3,-1])
print(tensor3)
```

```
tf.Tensor(
[[1. 1.]
 [1. 1.]
 [1. 1.]], shape=(3, 2), dtype=float32)
```

```
tf.compat.v1.disable_eager_execution()
```

```
a=tf.compat.v1.placeholder(dtype=tf.float32,shape=(20,20))
```

```
b=tf.compat.v1.placeholder(dtype=tf.float32,shape=(20,20))
```

perform mathematical operation on placeholder

[ ] ↳ *4 cells hidden*

Variable

[ ] ↳ *7 cells hidden*

Get Index of highest value in 2d matrix

```
tf.convert_to_tensor(t_2d)
```

```
<tf.Tensor 'ReadVariableOp:0' shape=(2, 2) dtype=int32>
```

```
t_2d=t_2d.assign([[2,2],[2,2]])
```

```
init = tf.compat.v1.global_variables_initializer()
with tf.compat.v1.Session() as sess:
  sess.run(init)
  ans = sess.run(t_2d)
  print(ans)
```

```
[[2 2]
 [2 2]]
```

```
x = tf.compat.v1.placeholder(tf.float32,shape = [5,10])
w = tf.compat.v1.placeholder(tf.float32,shape = [10,1])
b = tf.fill((5,1),-1.1)
```

```
x_data = np.random.randn(5,10)
w_data = np.random.randn(10,1)
```

```
wx = tf.matmul(x,w)
```

```
wxb = wx+b
```

```
s = tf.reduce_max(wxb)
```

```
with tf.compat.v1.Session() as sess:
  outs = sess.run(s,feed_dict={x:x_data,w:w_data})
  print(outs)
```

```
1.1226207
```

```
a = tf.constant(2)
b = tf.constant(5)

d = tf.add(a,b)
c = tf.multiply(a,b)
f = tf.add(d,c)
e = tf.subtract(d,c)
g = tf.divide(f,e)
```

```
with tf.compat.v1.Session() as sess:
  ans = sess.run(g)
  print(ans)
```

```
-5.666666666666667
```

```
ones = tf.ones(shape=[2,4])
```

```
ones
```

```
<tf.Tensor 'ones:0' shape=(2, 4) dtype=float32>
```

```
with tf.compat.v1.Session() as snss:
  an = snss.run(ones)
  print(an)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

```
```

tf_ones_likes

```
te = tf.constant([[1,2,3],[4,5,6]])
a = tf.ones_like(te)
```

```
with tf.compat.v1.Session() as snss:
  print(snss.run(a))
```

```
[[1 1 1]
 [1 1 1]]
```

casting

```
x = tf.constant([1,2,3],name='x',dtype = tf.float32)
print(x.dtype)
```

```
<dtype: 'float32'>
```

```
x = tf.cast(x,dtype = tf.int32)
print(x.dtype)
```

```
<dtype: 'int32'>
```

```
x_data = np.random.randn(2000,3)
w_real = [0.3,0.5,0.1]
b_real = -0.2

y_data =np.matmul(w_real,x_data.T)+b_real
```

```
y_data
```

```
array([ 0.63826839, -0.70675103,  0.65255999, ..., -0.52657195,
        0.06455494,  0.37922966])
```

Core Algorithm

```
#Load Dataset
import pandas as pd
dftrain = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/train.csv')#trai
dfeval = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/eval.csv')#test

y_train = dftrain.pop('survived')#popping column
y_eval = dfeval.pop('survived')
```

```
dftrain.shape
```

```
(627, 9)
```

```
y_train.shape
```

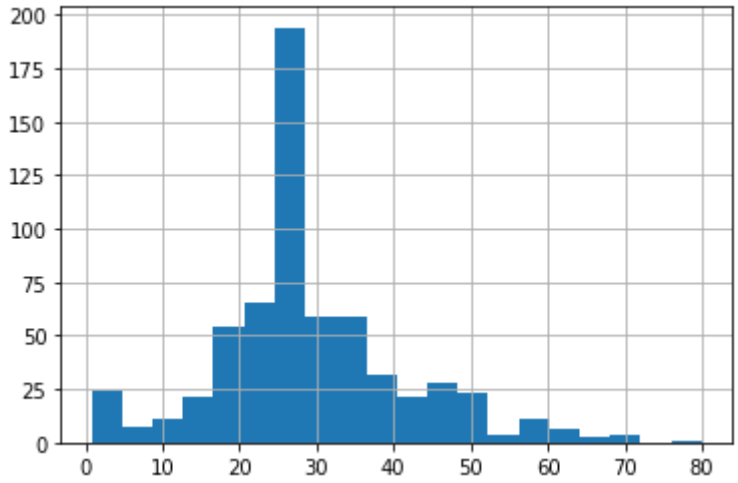
```
(627,)
```

```
dftrain.describe()
```

	age	n_siblings_spouses	parch	fare
count	627.000000	627.000000	627.000000	627.000000
mean	29.631308	0.545455	0.379585	34.385399
std	12.511818	1.151090	0.792999	54.597730
min	0.750000	0.000000	0.000000	0.000000
25%	23.000000	0.000000	0.000000	7.895800
50%	28.000000	0.000000	0.000000	15.045800
75%	35.000000	1.000000	0.000000	31.387500
max	80.000000	8.000000	5.000000	512.329200

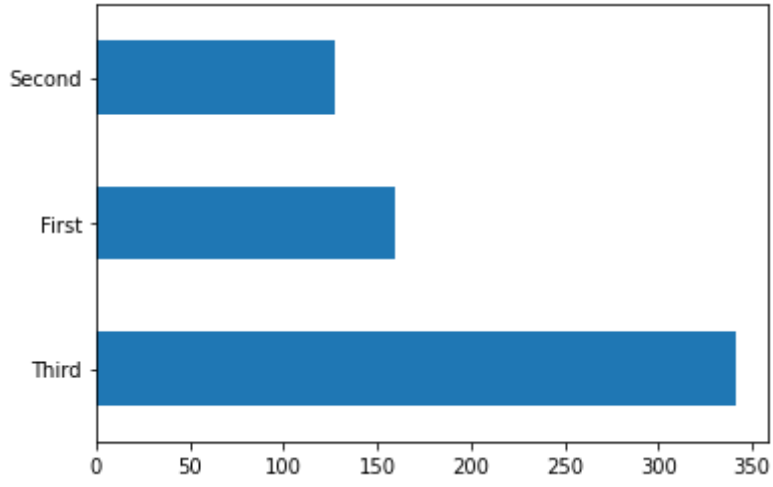
```
import matplotlib.pyplot as plt
dftrain.age.hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fea2c1f7910>
```




```
dftrain['class'].value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fea256efa10>
```



dealing with categorical data and real data in tf

```
dfeval.shape
```

```
(264, 9)
```

```
CATEGORICAL_COLUMNS = ['sex','n_siblings_spouses','parch','class','deck','embark_town','al
NUMERIC_COLUMNS = ['age','fare']

feature_columns = []
for feature_name in CATEGORICAL_COLUMNS:
  vocabulary = dftrain[feature_name].unique()
  feature_columns.append(tf.feature_column.categorical_column_with_vocabulary_list(feature_
  print(feature_columns)
for feature_name in NUMERIC_COLUMNS:
  feature_columns.append(tf.feature_column.numeric_column(feature_name,dtype = tf.float32))

#print(feature_columns)
```

```
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
[VocabularyListCategoricalColumn(key='sex', vocabulary_list=('male', 'female'), dtype
```

creating tf.data.Dataset

```
def make_input_fn(data_df, label_df, num_epochs=20, shuffle=True, batch_size=32):
  def input_function():
    ds = tf.data.Dataset.from_tensor_slices((dict(data_df), label_df))
    if shuffle:
      ds = ds.shuffle(1000)
    ds = ds.batch(batch_size).repeat(num_epochs)
    return ds
  return input_function

train_input_fn = make_input_fn(dftrain, y_train)
eval_input_fn = make_input_fn(dfeval, y_eval, num_epochs=1, shuffle=False)
```

```
linear_est = tf.estimator.LinearClassifier(feature_columns=feature_columns)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmpwq7xhr92
INFO:tensorflow:Using config: {'_model_dir': '/tmp/tmpwq7xhr92', '_tf_random_seed': N
```

```
  graph_options {
    rewrite_options {
      meta_optimizer_iterations: ONE
    }
  }
  , '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_cour
```

```
result = linear_est.evaluate(eval_input_fn)
```

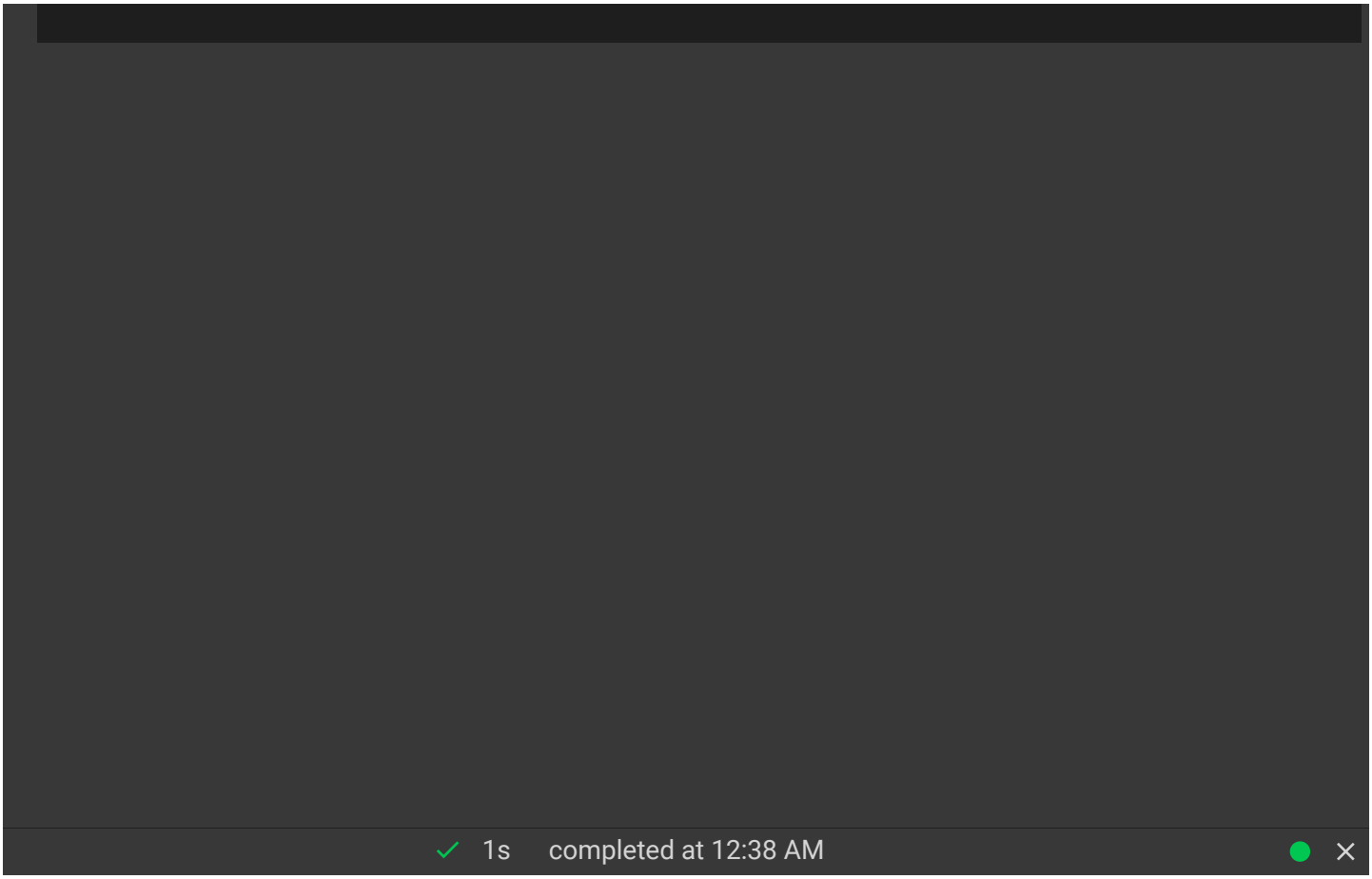
```
  INFO:tensorflow:Could not find trained model in model_dir: /tmp/tmpwq7xhr92, running
  INFO:tensorflow:Calling model_fn.
  /usr/local/lib/python3.7/dist-packages/tensorflow_estimator/python/estimator/canned/l
    getter=tf.compat.v1.get_variable)
  INFO:tensorflow:Done calling model_fn.
  INFO:tensorflow:Starting evaluation at 2022-05-07T19:04:20
  INFO:tensorflow:Graph was finalized.
  INFO:tensorflow:Running local_init_op.
  INFO:tensorflow:Done running local_init_op.
  INFO:tensorflow:Inference Time : 1.40364s
  INFO:tensorflow:Finished evaluation at 2022-05-07-19:04:22
  INFO:tensorflow:Saving dict for global step 0: accuracy = 0.625, accuracy_baseline =
```

```
print(result)
```

```
  {'accuracy': 0.625, 'accuracy_baseline': 0.625, 'auc': 0.5, 'auc_precision_recall': 0
```

```
res = list(linear_est.predict(eval_input_fn))
print(dfeval.loc[100])
print(res[100]['probabilities'])
```

```
  INFO:tensorflow:Could not find trained model in model_dir: /tmp/tmpwq7xhr92, running
  INFO:tensorflow:Calling model_fn.
  /usr/local/lib/python3.7/dist-packages/tensorflow_estimator/python/estimator/canned/l
    getter=tf.compat.v1.get_variable)
  INFO:tensorflow:Done calling model_fn.
  INFO:tensorflow:Graph was finalized.
  INFO:tensorflow:Running local_init_op.
  INFO:tensorflow:Done running local_init_op.
  sex                         male
  age                         30.0
  n_siblings_spouses             0
  parch                          0
  fare                        7.25
  class                      Third
  deck                     unknown
  embark_town          Southampton
  alone                          y
  Name: 100, dtype: object
  [0.5 0.5]
```


1s completed at 12:38 AM