compiler

September 6, 2021

[ ]:
```
from numpy.random import seed
seed(7567)
from tensorflow import set_random_seed
set_random_seed(7567)
```

[3]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from datagen import *
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth=True
sess = tf.compat.v1.Session(config=config)

NUM_ROWS = 20000
```

[2]:
```
#statistics collection
num_runs=50
num_rowss=[2500,5000,10000,20000,50000]
batch_size=32
epochss=[100,200]
TARS = tf.keras.models.Sequential()
TARS.add(tf.keras.layers.Dense(input_dim = 29 ,units = 50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=30, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=1, activation='linear'))
TARS.compile(optimizer="adam", loss="mean_squared_error", metrics=['mse'])

matrix=[]
for epochs in epochss:
    epoch_row=[]
    for num_rows in num_rowss:
        loss_array=[]
```

```
        for i in range(num_runs):
            dframe = pd.DataFrame()
            dframe = datagen(num_rows)
            x = dframe.iloc[:, 1:-1].values
            y = dframe.iloc[:, -1].values
            le = LabelEncoder()
            x[:, 0] = le.fit_transform(x[:,0])
            ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[2])], remainder='passthrough')
            x = np.array(ct.fit_transform(x))
            ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[7])], remainder='passthrough')
            x = np.array(ct2.fit_transform(x))
            ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[18])], remainder='passthrough')
            x = np.array(ct3.fit_transform(x))
            scalarX, scalarY = MinMaxScaler(), MinMaxScaler()
            scalarX.fit(x)
            scalarY.fit(y.reshape(num_rows,1))
            x = scalarX.transform(x)
            y = scalarY.transform(y.reshape(num_rows,1))
            x_train, x_test, y_train, y_test = train_test_split(x, y, test_size␣
↪= 0.2)
            for ix, layer in enumerate(TARS.layers):
                if hasattr(TARS.layers[ix], 'kernel_initializer') and␣
↪hasattr(TARS.layers[ix], 'bias_initializer'):
                    weight_initializer=TARS.layers[ix].kernel_initializer
                    bias_initializer=TARS.layers[ix].bias_initializer

                    old_weights, old_biases = TARS.layers[ix].get_weights()

                    TARS.layers[ix].
↪set_weights([weight_initializer(shape=old_weights.shape),␣
↪bias_initializer(shape=len(old_biases))])
            TARS.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,␣
↪verbose=0)
            error=TARS.evaluate(x_test, y_test, verbose=0)[0]
            #print(f"Run {i} finished, mean_squared_error: {error}")
            y_pred = TARS.predict(x_test)
            y_test = scalarY.inverse_transform(y_test)
            y_pred = scalarY.inverse_transform(y_pred)
            explained_variance = 1 - np.var(y_test - y_pred)/np.var(y_test)
            mse = (sum((y_test-y_pred)**2)/len(y_pred))[0]
            loss_array.append([error, explained_variance, mse])
```

```
        print(f"Epoch number: {epochs}, num_rows: {num_rows}, model loss:␣
↪{sum([a[0] for a in loss_array])/len(loss_array)}, explained variance:␣
↪{sum([a[1] for a in loss_array])/len(loss_array)*100}%, mse: {sum([a[2] for␣
↪a in loss_array])/len(loss_array)}")
        epoch_row.append(loss_array)
    matrix.append(epoch_row)
print(matrix)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-2-72805c8bab71> in <module>
     43
     44                     TARS.layers[ix].
↪set_weights([weight_initializer(shape=old_weights.shape),␣
↪bias_initializer(shape=len(old_biases))])
---> 45             TARS.fit(x_train, y_train, batch_size=batch_size,␣
↪epochs=epochs, verbose=0)
     46             error=TARS.evaluate(x_test, y_test, verbose=0)[0]
     47             #print(f"Run {i} finished, mean_squared_error: {error}")

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\training.
↪py in fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
↪validation_freq, max_queue_size, workers, use_multiprocessing)
   1181                 _r=1):
   1182               callbacks.on_train_batch_begin(step)
-> 1183               tmp_logs = self.train_function(iterator)
   1184               if data_handler.should_sync:
   1185                 context.async_wait()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.
↪py in __call__(self, *args, **kwds)
    887
    888       with OptionalXlaContext(self._jit_compile):
--> 889         result = self._call(*args, **kwds)
    890
    891       new_tracing_count = self.experimental_get_tracing_count()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.
↪py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we
↪run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint:␣
↪disable=not-callable
    918     elif self._stateful_fn is not None:
```

```
    919       # Release the lock early so that multiple threads can perform the
 ↪call

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio
 ↪py in __call__(self, *args, **kwargs)
   3021       (graph_function,
   3022        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3023     return graph_function._call_flat(
   3024         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
   3025

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio
 ↪py in _call_flat(self, args, captured_inputs, cancellation_manager)
   1958         and executing_eagerly):
   1959       # No tape is watching; skip to running the function.
-> 1960       return self._build_call_outputs(self._inference_function.call(
   1961           ctx, args, cancellation_manager=cancellation_manager))
   1962     forward_backward = self._select_forward_and_backward_functions(

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio
 ↪py in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
--> 591           outputs = execute.execute(
    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\execute
 ↪py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

[4]:
```
#statistics collection
num_runs=1
num_rowss=[50000]
batch_size=32
epochss=[25]
TARS = tf.keras.models.Sequential()
```

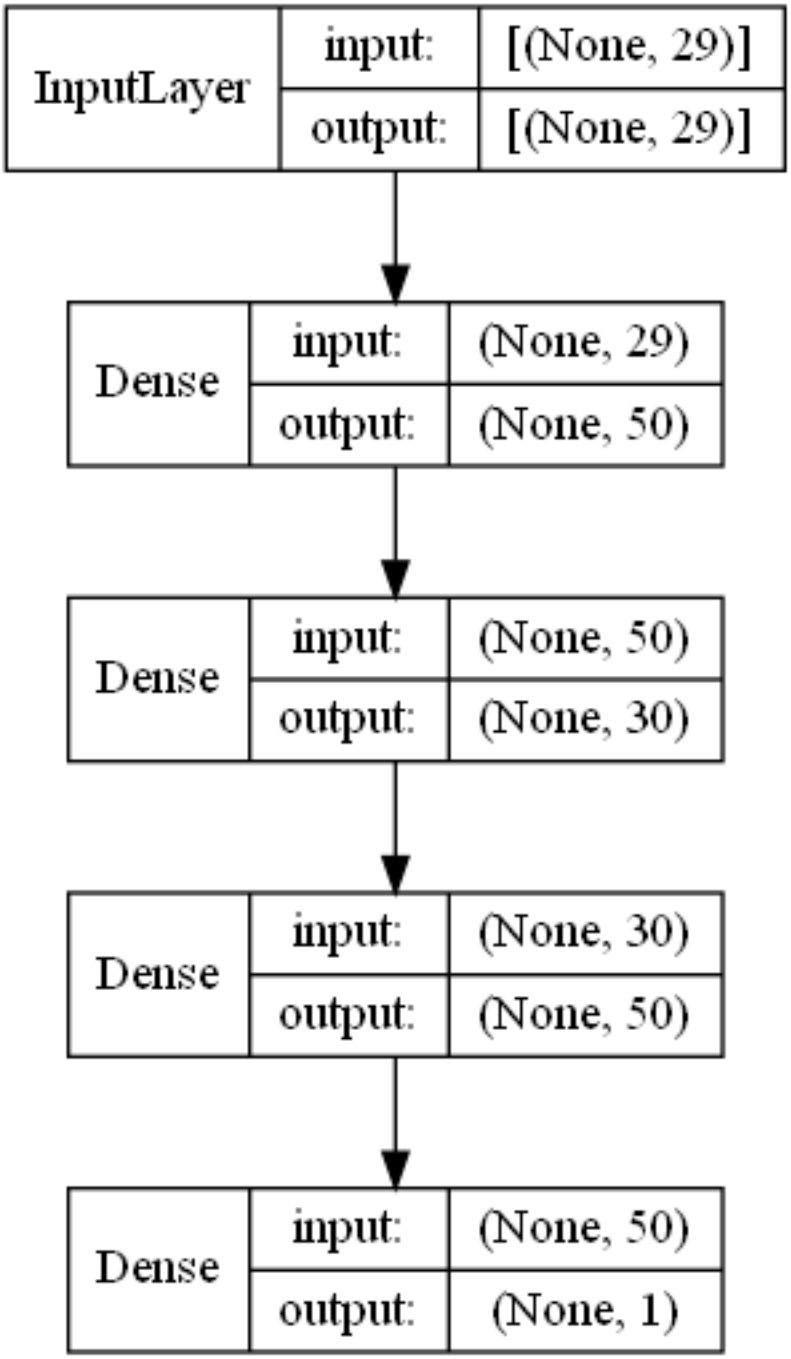
```
TARS.add(tf.keras.layers.Dense(input_dim = 29 ,units = 50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=30, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=1, activation='linear'))
TARS.compile(optimizer="adam", loss="mean_squared_error", metrics=['mse'])

matrix=[]
for epochs in epochss:
    epoch_row=[]
    for num_rows in num_rowss:
        loss_array=[]
        for i in range(num_runs):
            dframe = pd.DataFrame()
            dframe = datagen(num_rows)
            x = dframe.iloc[:, 1:-1].values
            y = dframe.iloc[:, -1].values
            le = LabelEncoder()
            x[:, 0] = le.fit_transform(x[:,0])
            ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[2])], remainder='passthrough')
            x = np.array(ct.fit_transform(x))
            ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[7])], remainder='passthrough')
            x = np.array(ct2.fit_transform(x))
            ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),␣
↪[18])], remainder='passthrough')
            x = np.array(ct3.fit_transform(x))
            scalarX, scalarY = MinMaxScaler(), MinMaxScaler()
            scalarX.fit(x)
            scalarY.fit(y.reshape(num_rows,1))
            x = scalarX.transform(x)
            y = scalarY.transform(y.reshape(num_rows,1))
            x_train, x_test, y_train, y_test = train_test_split(x, y, test_size␣
↪= 0.2)
            for ix, layer in enumerate(TARS.layers):
                if hasattr(TARS.layers[ix], 'kernel_initializer') and␣
↪hasattr(TARS.layers[ix], 'bias_initializer'):
                    weight_initializer=TARS.layers[ix].kernel_initializer
                    bias_initializer=TARS.layers[ix].bias_initializer

                    old_weights, old_biases = TARS.layers[ix].get_weights()

                    TARS.layers[ix].
↪set_weights([weight_initializer(shape=old_weights.shape),␣
↪bias_initializer(shape=len(old_biases))])
```

```
            TARS.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,␣
 ↪verbose=0)
            error=TARS.evaluate(x_test, y_test, verbose=0)[0]
            #print(f"Run {i} finished, mean_squared_error: {error}")
            y_pred = TARS.predict(x_test)
            y_test = scalarY.inverse_transform(y_test)
            y_pred = scalarY.inverse_transform(y_pred)
            explained_variance = 1 - np.var(y_test - y_pred)/np.var(y_test)
            mse = (sum((y_test-y_pred)**2)/len(y_pred))[0]
            loss_array.append([error, explained_variance, mse])
        print(f"Epoch number: {epochs}, num_rows: {num_rows}, model loss:␣
 ↪{sum([a[0] for a in loss_array])/len(loss_array)}, explained variance:␣
 ↪{sum([a[1] for a in loss_array])/len(loss_array)*100}%, mse: {sum([a[2] for␣
 ↪a in loss_array])/len(loss_array)}")
        epoch_row.append(loss_array)
    matrix.append(epoch_row)
print(matrix)
```

```
Epoch number: 25, num_rows: 50000, model loss: 0.0031147398985922337, explained
variance: 86.7621646415881%, mse: 0.13374402977212801
[[[[0.0031147398985922337, 0.867621646415881, 0.13374402977212801]]]]
```

[5]:
```
tf.keras.utils.plot_model(
    TARS,
    to_file="model.png",
    show_shapes=True,
    show_dtype=False,
    show_layer_names=False,
    rankdir="TB",
    expand_nested=True,
    dpi=96,
)
```

[5]:



[5]:
```
from ann_visualizer.visualize import ann_viz

ann_viz(TARS, title="Keras Model Diagram")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-5-6635458f7d32> in <module>
      1 from ann_visualizer.visualize import ann_viz
      2
----> 3 ann_viz(TARS, title="Keras Model Diagram")

~\anaconda3\lib\site-packages\ann_visualizer\visualize.py in ann_viz(model,␣
↪view, filename, title)
    121                 c.node(str(n), label="Image\n"+pxls[1]+" x"+pxls[2]+"␣
↪pixels\n"+clrmap, fontcolor="white");
    122             else:
--> 123                 raise ValueError("ANN Visualizer: Layer not supported␣
↪for visualizing");
    124         for i in range(0, hidden_layers_nr):
    125             with g.subgraph(name="cluster_"+str(i+1)) as c:

ValueError: ANN Visualizer: Layer not supported for visualizing
```

[2]:
```
dframe = pd.DataFrame()
dframe = datagen(NUM_ROWS)
```

[3]:
```
dframe
```

[3]:

	studentIDs	gender	age	teacher_cred	class_size	\
0	21097	Female	10	Associate's	25	
1	31103	Male	13	Associate's	32	
2	23138	Male	9	Master's	20	
3	35902	Male	14	Bachelor's	25	
4	23691	Male	18	Bachelor's	27	
…	…	…	…	…	…	
19995	15635	Female	14	Master's	36	
19996	37345	Female	18	Bachelor's	28	
19997	31572	Male	18	Associate's	33	
19998	46247	Male	13	PhD	28	
19999	10361	Female	12	Bachelor's	35	

	disability	accomadation	gpadifference
0	Auditory Disability	Special Education Classroom	-1.903908
1	Mathematics Disability	Book Buddy	0.495376
2	Mathematics Disability	Tutoring Sessions	0.730660
3	Auditory Disability	Breakout Corner	1.200479
4	Auditory Disability	Materials in Braille	1.589534
…	…	…	…
19995	Developmentally Delayed	Breakout Corner	-1.707976
19996	Visual Disability	Text to Speech Devices	0.954426
19997	Auditory Disability	Use of Calculator on Tests	-1.108240
19998	Dyslexia	Use of Calculator on Tests	1.057671
19999	Visual Disability	Tutoring Sessions	-0.372247

```
[20000 rows x 8 columns]
```

[4]:
```
x = dframe.iloc[:, 1:-1].values
y = dframe.iloc[:, -1].values
```

[5]:
```
le = LabelEncoder()
x[:, 0] = le.fit_transform(x[:,0])
print(x)
```

```
[[1 16 "Bachelor's" 27 'Autism' 'Breakout Corner']
 [0 15 "Bachelor's" 34 'Developmentally Delayed' 'Bigger Print Materials']
 [0 18 'PhD' 40 'Visual Disability' 'Use of Toy in Class']
 …
 [1 7 'PhD' 32 'Autism' 'Book Buddy']
 [1 11 "Associate's" 31 'Down Syndrome' 'Bigger Print Materials']
 [0 14 "Associate's" 34 'Developmentally Delayed' 'Isolated Workstation']]
```

[6]:
```
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [2])],␣
 ↪remainder='passthrough')
x = np.array(ct.fit_transform(x))
ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [7])],␣
 ↪remainder='passthrough')
x = np.array(ct2.fit_transform(x))
ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [18])],␣
 ↪remainder='passthrough')
x = np.array(ct3.fit_transform(x))
```

[7]:
```
scalarX, scalarY = MinMaxScaler(), MinMaxScaler()
scalarX.fit(x)
scalarY.fit(y.reshape(NUM_ROWS,1))
x = scalarX.transform(x)
y = scalarY.transform(y.reshape(NUM_ROWS,1))
```

[8]:
```
# Coulums 0-3 are highest degree
# colums 4 is gender
# colum 5 is normalized age
# colum 6 is normalized class sized
```

[9]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,␣
 ↪random_state=7687)
```

[10]:
```
# container for adding layers to nuerual net
TARS = tf.keras.models.Sequential()
```

[11]:
```
TARS.add(tf.keras.layers.Dense(input_dim = 29 ,units = 50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=30, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=50, activation='relu'))
```

```
TARS.add(tf.keras.layers.Dense(units=1, activation='linear'))
```

[12]:
```
TARS.compile(optimizer="adam", loss="mean_squared_error", metrics=['mse'])
```

[13]:
```
TARS.fit(x_train, y_train, batch_size=32, epochs=100)
```

```
Epoch 1/100
500/500 [==============================] - 2s 2ms/step - loss: 0.0199 - mse:
0.0199
Epoch 2/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0053 - mse:
0.0053
Epoch 3/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0046 - mse:
0.0046
Epoch 4/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0045 - mse:
0.0045
Epoch 5/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0044 - mse:
0.0044
Epoch 6/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0043 - mse:
0.0043
Epoch 7/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0043 - mse:
0.0043
Epoch 8/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0042 - mse:
0.0042
Epoch 9/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0041 - mse:
0.0041
Epoch 10/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0041 - mse:
0.0041
Epoch 11/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0041 - mse:
0.0041
Epoch 12/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0041 - mse:
0.0041
Epoch 13/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0041 - mse:
0.0041
Epoch 14/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0040 - mse:
0.0040
```

```
Epoch 15/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0040 - mse:
0.0040
Epoch 16/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0040 - mse:
0.0040
Epoch 17/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0040 - mse:
0.0040
Epoch 18/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0040 - mse:
0.0040
Epoch 19/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0039 - mse:
0.0039
Epoch 20/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0039 - mse:
0.0039
Epoch 21/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0039 - mse:
0.0039
Epoch 22/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0039 - mse:
0.0039
Epoch 23/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0039 - mse:
0.0039
Epoch 24/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0039 - mse:
0.0039
Epoch 25/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0038 - mse:
0.0038
Epoch 26/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0038 - mse:
0.0038
Epoch 27/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0038 - mse:
0.0038
Epoch 28/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0038 - mse:
0.0038
Epoch 29/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 30/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
```

```
Epoch 31/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0038 - mse:
0.0038
Epoch 32/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 33/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 34/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 35/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 36/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 37/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 38/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0037 - mse:
0.0037
Epoch 39/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 40/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 41/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 42/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 43/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 44/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 45/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 46/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
```

```
Epoch 47/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0036 - mse:
0.0036
Epoch 48/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 49/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 50/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 51/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 52/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 53/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 54/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 55/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 56/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 57/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 58/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 59/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 60/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 61/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 62/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
```

```
Epoch 63/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 64/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0035 - mse:
0.0035
Epoch 65/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 66/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 67/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 68/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 69/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 70/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 71/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 72/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 73/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 74/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 75/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
Epoch 76/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 77/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034A: 0s - loss: 0.0033 - mse
Epoch 78/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0034 - mse:
0.0034
```

```
Epoch 79/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 80/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 81/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 82/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 83/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 84/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 85/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 86/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 87/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 88/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 89/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 90/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 91/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 92/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 93/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 94/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
```

```
Epoch 95/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 96/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 97/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0032 - mse:
0.0032
Epoch 98/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 99/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0033 - mse:
0.0033
Epoch 100/100
500/500 [==============================] - 1s 2ms/step - loss: 0.0032 - mse:
0.0032
```

[13]:
```
<tensorflow.python.keras.callbacks.History at 0x24506243070>
```

- List item
- List item

[14]:
```
TARS.save("weights.h5")
```

[15]:
```
TARS.evaluate(x_test, y_test)
```

```
125/125 [==============================] - 0s 968us/step - loss: 0.0044 - mse:
0.0044
```

[15]:
```
[0.004357358906418085, 0.004357358906418085]
```

[19]:
```
TARS.predict(x_test)
```

[19]:
```
array([[0.38861433],
       [0.8379196 ],
       [0.61360025],
       …,
       [0.5294008 ],
       [0.43504548],
       [0.51618   ]], dtype=float32)
```

[18]:
```
x_test[0:1]
```

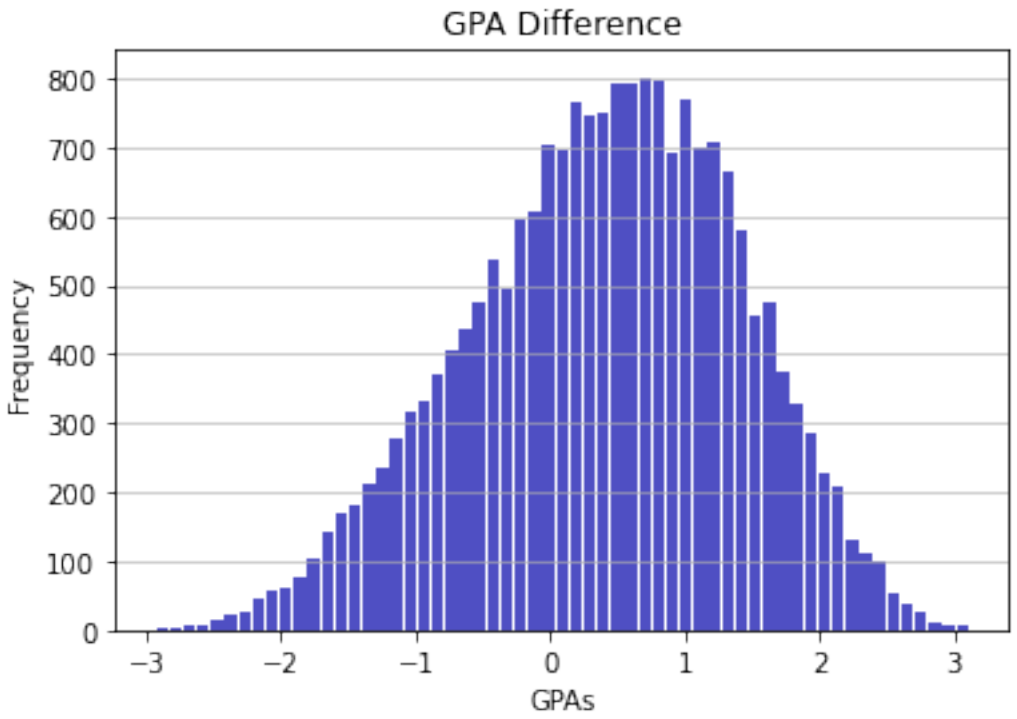
[18]:
```
array([[0.        , 0.        , 0.        , 1.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 1.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
        0.        , 0.        , 0.        , 0.        , 1.        ,
        0.        , 1.        , 0.66666667, 0.48717949]])
```

[38]:
```
# give me a student and lets try some accomedation
# take in a normal row input
# changing accomedation
# class size
# teacher_cre
```

[4]:
```
n, bins, patches = plt.hist(x=dframe['gpadifference'], bins='auto',␣
 ↪color='#0504aa',
                            alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.xlabel('GPAs')
plt.ylabel('Frequency')
plt.title('GPA Difference')
```

[4]:
```
Text(0.5, 1.0, 'GPA Difference')
```



[1]:
```
scalarY.inverse_transform(TARS.predict(np.array([[0,0,0,1,1,0.5,0.5]])))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
<ipython-input-1-9eeeaf545e05> in <module>
----> 1 scalarY.inverse_transform(TARS.predict(np.array([[0,0,0,1,1,0.5,0.5]])))

NameError: name 'scalarY' is not defined
```

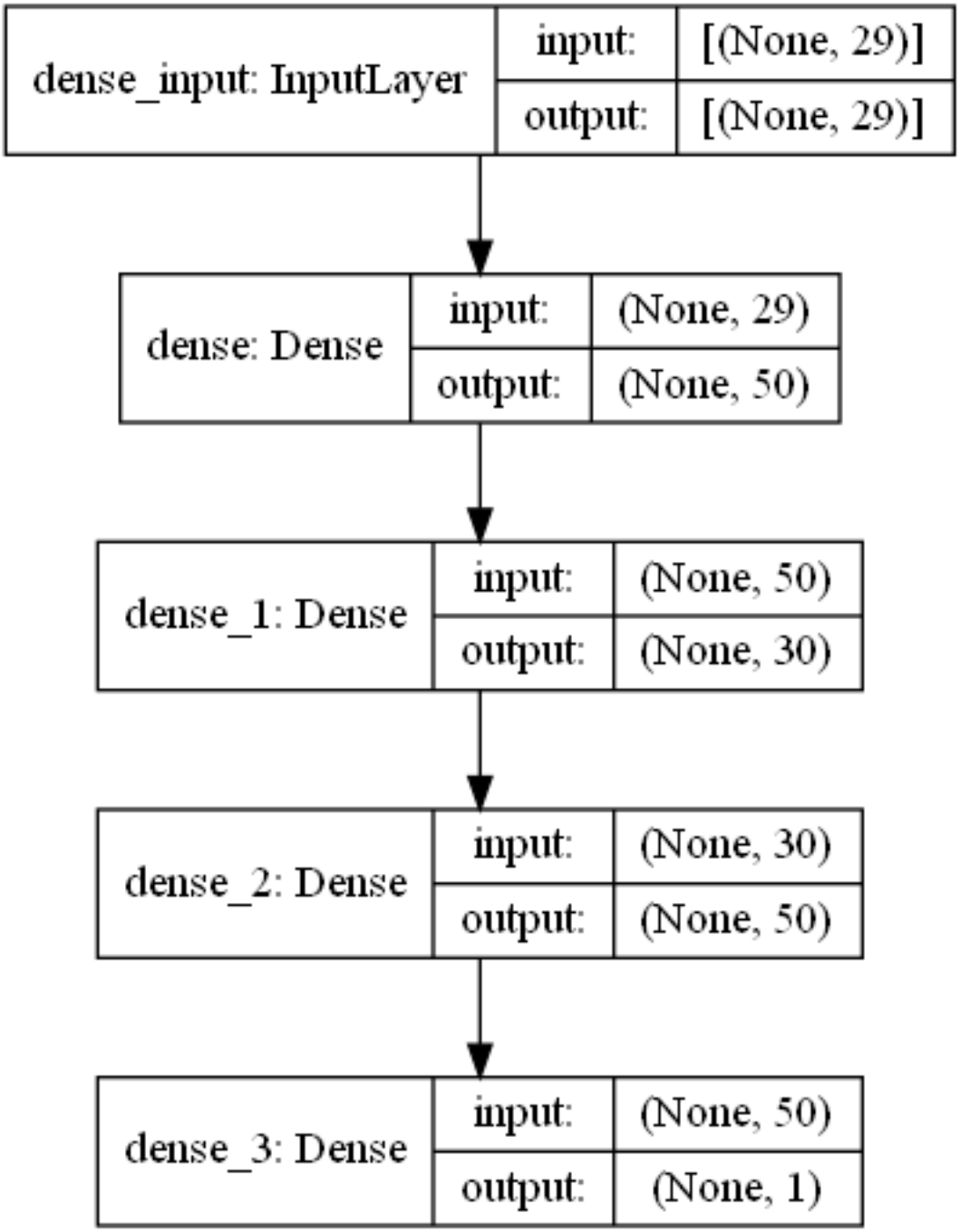
[ ]:
```
TARS.predict(np.array([[0,1,0,0,1,0.5,0.5]]))
```

1 Create the GridSearch estimator along with a parameter object containing the values to adjust

from sklearn.model_selection import GridSearchCV param_grid = {'C': [1, 5, 10, 50], 'gamma': [0.0001, 0.0005, 0.001, 0.005]} grid = GridSearchCV(model, param_grid, verbose=3) Mohan to Everyone (12:01 PM) grid.fit(X_train, y_train) print(grid.best_params_)

[2]:
```
TARS = tf.keras.models.Sequential()
TARS.add(tf.keras.layers.Dense(input_dim = 29 ,units = 50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=30, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=50, activation='relu'))
TARS.add(tf.keras.layers.Dense(units=1, activation='linear'))
TARS.compile(optimizer="adam", loss="mean_squared_error", metrics=['mse',␣
↪'accuracy'])
TARS.load_weights('weights.h5')
from keras.utils.vis_utils import plot_model
plot_model(TARS, to_file='model_plot.png', show_shapes=True,␣
↪show_layer_names=True)
```

[2]:

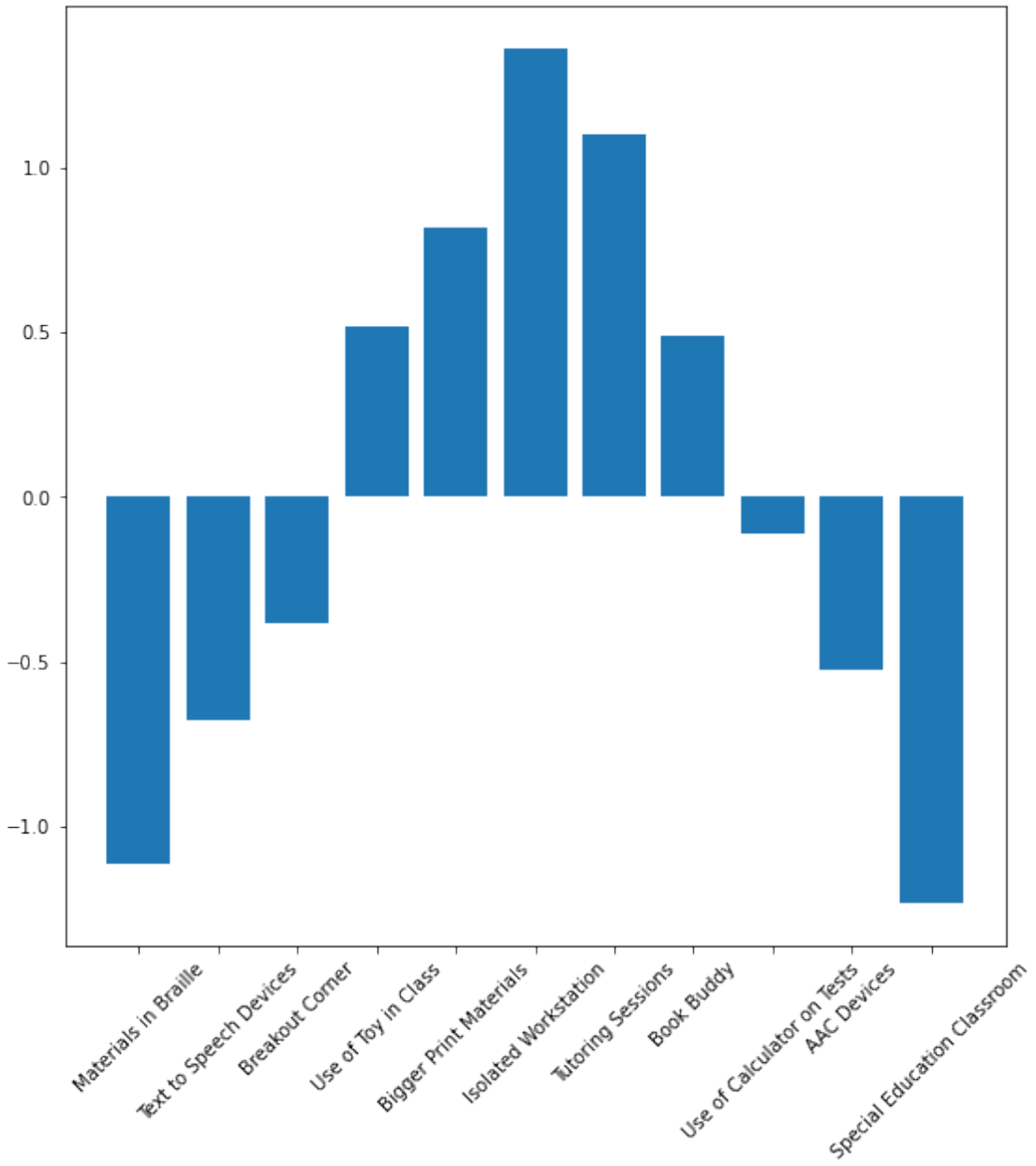


[17]:
```
def predict(sample_student):
    maxdif = -4
    bestaccom = ""
```

```
    accomodation_list = ["Materials in Braille", "Text to Speech Devices",␣
↪"Breakout Corner", "Use of Toy in Class", "Bigger Print Materials",␣
↪"Isolated Workstation", "Tutoring Sessions", "Book Buddy", "Use of␣
↪Calculator on Tests", "AAC Devices", "Special Education Classroom"]
    gpadiffs = []
    for i in accomodation_list:
        temparray= [[]]
        temparray[0] = np.append(sample_student, i)
        temparray = np.array(temparray)
        temparray[:, 0] = le.transform(temparray[:,0])
        temparray = np.array(ct.transform(temparray))
        temparray = np.array(ct2.transform(temparray))
        temparray = np.array(ct3.transform(temparray))
        temparray = scalarX.transform(temparray)
        gpadiffs.append(scalarY.inverse_transform(TARS.predict(np.
↪array(temparray)))[0][0])
        if (scalarY.inverse_transform(TARS.predict(np.array(temparray))) >␣
↪maxdif):
            maxdif = scalarY.inverse_transform(TARS.predict(np.
↪array(temparray)))
            bestaccom = i
    fig = plt.figure(figsize=(7,7))
    ax = fig.add_axes([0,0,1,1])
    plt.xticks(rotation=45)
    ax.bar(accomodation_list, gpadiffs)
    return(bestaccom + " is the predicted best accomadation.")
```

[19]:
```
#predicting which accommodation is most effective given a student
#input: gender, age, teacher_cred, class size, disability
predict(np.array([["Female", "15", "Bachelor's", "25", "ADHD"]]))
```

[19]: 'Isolated Workstation is the predicted best accomadation.'



[ ]:
```
#TARS = tf.keras.models.load_model('weights.h5')
```

[ ]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [1, 5, 10, 50], 'gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(TARS, param_grid, verbose=3)
```

[ ]:
```
# TARS.layers[1].get_weights()
```

GridSearch Figure out which columns are the most impactful transform outputs back fix environment

[ ]:
```
# from keras.wrappers.scikit_learn import KerasClassifier, KerasRegressor
# import eli5
# from eli5.sklearn import PermutationImportance
```

[ ]:
```
# my_model = KerasRegressor(build, epochs=100, batch_size=200)
# my_model.fit(x,y)
```

[ ]:
```
# from sklearn.model_selection import KFold
# from sklearn.model_selection import cross_val_score

# kfold = KFold(n_splits=10)
# results = cross_val_score(my_model, x, y, cv=kfold)
```

[ ]:
```
# TARS
```

[ ]:
```
# import shap
```

[ ]:
```
# shap.initjs()
```

[ ]:
```
# explainer = shap.TreeExplainer(TARS)
# shap_values = explainer.shap_values(x)
```

[ ]:
```
# # permutation feature importance with knn for classification
# from sklearn.datasets import make_classification
# from sklearn.neighbors import KNeighborsClassifier
# from sklearn.inspection import permutation_importance
# from matplotlib import pyplot
# # define dataset
# # define the model
# # fit the model
# # perform permutation importance
# results = permutation_importance(TARS, x, y, scoring='accuracy')
# # get importance
# importance = results.importances_mean
# # summarize feature importance
# for i,v in enumerate(importance):
#         print('Feature: %0d, Score: %.5f' % (i,v))
# # plot feature importance
# pyplot.bar([x for x in range(len(importance))], importance)
# pyplot.show()
```

[ ]:
```

```