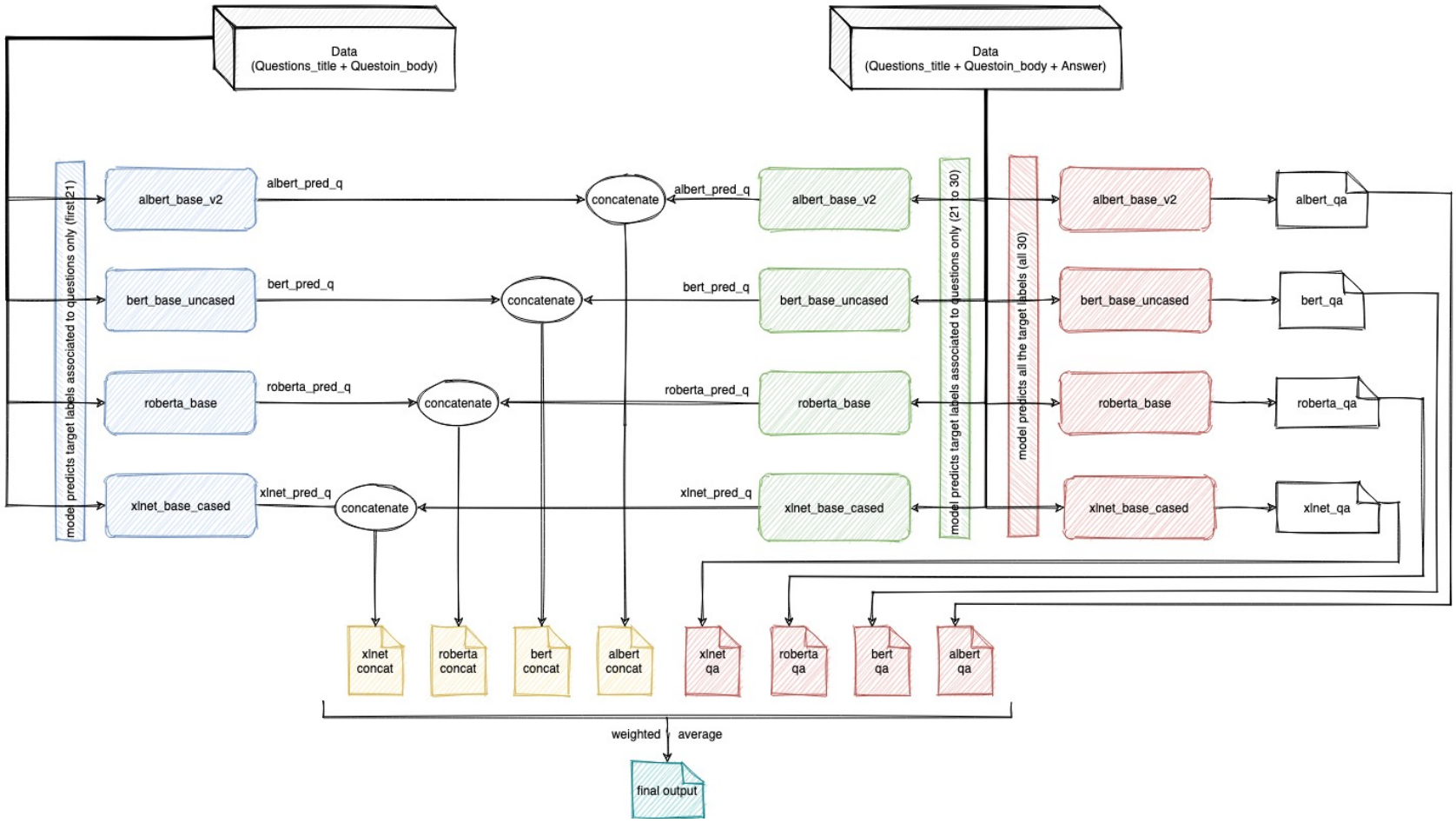
In [ ]:
```
# importing necessary libraries
import numpy as np
import pandas as pd
import tensorflow as tf
```

In [27]:
```
# Ensemble of transformer architecture
%matplotlib inline
from IPython.display import Image
Image('diagram_ensemble.jpg')
```

Out[27]:



In [ ]:
```
!wget 'https://github.com/SarthakV7/Kaggle_google_quest_challenge/blob/master/transformers_data.zip?raw=true'
!mv 'transformers_data.zip?raw=true' 'transformers_data.zip'
!unzip transformers_data
```

In [ ]:
```
# performance comparasion of all the transformers used

from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["model","dataset","train loss","cv loss","train rhos","cv rhos"]
x.add_row(['bert_base_uncased','questions',0.3393, 0.3302, 0.5543, 0.6013])
x.add_row(['bert_base_uncased','answer',0.3320, 0.3278, 0.4967, 0.5438])
x.add_row(['bert_base_uncased','question+answer',0.3287, 0.3166, 0.5511, 0.6109])

x.add_row(['roberta_base','questions',0.3542, 0.3400, 0.4953, 0.5674])
x.add_row(['roberta_base','answer',0.3430, 0.3253, 0.3927, 0.4993])
x.add_row(['roberta_base','question+answer',0.3546, 0.3397, 0.4305, 0.5082])

x.add_row(['albert_base_v2','questions',0.4022, 0.3785, 0.3252, 0.4341])
x.add_row(['albert_base_v2','answer',0.3829, 0.3660, 0.2278, 0.3438])
x.add_row(['albert_base_v2','question+answer',0.4046, 0.3790, 0.2619, 0.3680])

x.add_row(['xlnet_base_cased','questions',0.3662, 0.3412, 0.4679, 0.5685])
x.add_row(['xlnet_base_cased','answer',0.3611, 0.3401, 0.3531, 0.4702])
x.add_row(['xlnet_base_cased','question+answer',0.3721, 0.3452, 0.3942, 0.5013])
print(x)
```

```
+------------------+-----------------+------------+---------+------------+---------+
|      model       |     dataset     | train loss | cv loss | train rhos | cv rhos |
+------------------+-----------------+------------+---------+------------+---------+
| bert_base_uncased |    questions    |   0.3393   |  0.3302 |   0.5543   |  0.6013 |
| bert_base_uncased |      answer     |   0.332    |  0.3278 |   0.4967   |  0.5438 |
| bert_base_uncased | question+answer |   0.3287   |  0.3166 |   0.5511   |  0.6109 |
|    roberta_base   |    questions    |   0.3542   |   0.34  |   0.4953   |  0.5674 |
|    roberta_base   |      answer     |   0.343    |  0.3253 |   0.3927   |  0.4993 |
|    roberta_base   | question+answer |   0.3546   |  0.3397 |   0.4305   |  0.5082 |
|   albert_base_v2  |    questions    |   0.4022   |  0.3785 |   0.3252   |  0.4341 |
|   albert_base_v2  |      answer     |   0.3829   |  0.366  |   0.2278   |  0.3438 |
|   albert_base_v2  | question+answer |   0.4046   |  0.379  |   0.2619   |  0.368  |
|  xlnet_base_cased |    questions    |   0.3662   |  0.3412 |   0.4679   |  0.5685 |
|  xlnet_base_cased |      answer     |   0.3611   |  0.3401 |   0.3531   |  0.4702 |
|  xlnet_base_cased | question+answer |   0.3721   |  0.3452 |   0.3942   |  0.5013 |
+------------------+-----------------+------------+---------+------------+---------+
```

In [ ]:
```
# Importing the predicted labels of albert
albert_train_q = pd.read_csv('transformers_data/albert_pred_q_train.csv')
albert_test_q = pd.read_csv('transformers_data/albert_pred_q_test.csv')
albert_train_a = pd.read_csv('transformers_data/albert_pred_a_train.csv')
albert_test_a = pd.read_csv('transformers_data/albert_pred_a_test.csv')
albert_train_qa = pd.read_csv('transformers_data/albert_pred_qa_train.csv')
albert_test_qa = pd.read_csv('transformers_data/albert_pred_qa_test.csv')

albert_train_concat = pd.concat([albert_train_q, albert_train_a], axis=1)
albert_test_concat = pd.concat([albert_test_q, albert_test_a], axis=1)
```

In [ ]:
```
# Importing the predicted labels of bert
bert_train_q = pd.read_csv('transformers_data/bert_pred_q_train.csv')
bert_test_q = pd.read_csv('transformers_data/bert_pred_q_test.csv')
bert_train_a = pd.read_csv('transformers_data/bert_pred_a_train.csv')
bert_test_a = pd.read_csv('transformers_data/bert_pred_a_test.csv')
bert_train_qa = pd.read_csv('transformers_data/bert_pred_qa_train.csv')
bert_test_qa = pd.read_csv('transformers_data/bert_pred_qa_test.csv')

bert_train_concat = pd.concat([bert_train_q, bert_train_a], axis=1)
bert_test_concat = pd.concat([bert_test_q, bert_test_a], axis=1)
```

In [ ]:
```
# Importing the predicted labels of roberta
roberta_train_q = pd.read_csv('transformers_data/roberta_pred_q_train.csv')
roberta_test_q = pd.read_csv('transformers_data/roberta_pred_q_test.csv')
roberta_train_a = pd.read_csv('transformers_data/roberta_pred_a_train.csv')
roberta_test_a = pd.read_csv('transformers_data/roberta_pred_a_test.csv')
roberta_train_qa = pd.read_csv('transformers_data/roberta_pred_qa_train.csv')
roberta_test_qa = pd.read_csv('transformers_data/roberta_pred_qa_test.csv')

roberta_train_concat = pd.concat([roberta_train_q, roberta_train_a], axis=1)
roberta_test_concat = pd.concat([roberta_test_q, roberta_test_a], axis=1)
```

In [ ]:
```
# Importing the predicted labels of xlnet
xlnet_train_q = pd.read_csv('transformers_data/xlnet_pred_q_train.csv')
xlnet_test_q = pd.read_csv('transformers_data/xlnet_pred_q_test.csv')
xlnet_train_a = pd.read_csv('transformers_data/xlnet_pred_a_train.csv')
xlnet_test_a = pd.read_csv('transformers_data/xlnet_pred_a_test.csv')
xlnet_train_qa = pd.read_csv('transformers_data/xlnet_pred_qa_train.csv')
xlnet_test_qa = pd.read_csv('transformers_data/xlnet_pred_qa_test.csv')

xlnet_train_concat = pd.concat([xlnet_train_q, xlnet_train_a], axis=1)
xlnet_test_concat = pd.concat([xlnet_test_q, xlnet_test_a], axis=1)
```

In [ ]:
```
# Importing the test data and sample submission data
train = pd.read_csv('transformers_data/train.csv')
test = pd.read_csv('transformers_data/test.csv')
submission = pd.read_csv('transformers_data/sample_submission.csv')
```

In [ ]:
```
# For binning, I've used the below code from:
# https://www.kaggle.com/markpeng/ensemble-5models-v4-v7-magic/notebook?select=submission.csv#Do-Inference
'''
Here the author has created 60 bins that correspond to 60 euqally spaced percentile values (between 1-100)
of the 25 distinct target labels. The idea is to take the predicted values as an input and then preprocess
them such that the final values are all from the 60 bins. This helps in making the predicted data much more
structured/ordered.
'''

X = pd.read_csv('transformers_data/train.csv').iloc[:, 11:]
unique_labels = np.unique(X.values)
denominator = 60
q = np.arange(0, 101, 100 / denominator)
exp_labels = np.percentile(unique_labels, q) # Generating the 60 bins.

def optimize_ranks(preds, unique_labels=exp_labels):
    new_preds = np.zeros(preds.shape)
    for i in range(preds.shape[1]):
        interpolate_bins = np.digitize(preds[:, i],
                                       bins=unique_labels,
                                       right=False)

        if len(np.unique(interpolate_bins)) == 1:
            new_preds[:, i] = preds[:, i]
        else:
            new_preds[:, i] = unique_labels[interpolate_bins]

    return new_preds
```

In [ ]:
```
# importing tensorflowgraph elements
from tensorflow.keras.layers import Input, Dense, Dropout, Concatenate
from tensorflow.keras.models import Model
import tensorflow.keras.backend as K
```

In [ ]:
```
# Function for creating a final model that takes the labels predicted by transformers as input and ge
nerates output
def create_model():
  K.clear_session()

  albert_q_layer = Input(21, name='albert_q_layer', dtype=tf.float32)
  albert_a_layer = Input(9, name='albert_a_layer', dtype=tf.float32)
  albert_qa_layer = Input(30, name='albert_qa_layer', dtype=tf.float32)
  albert_concat_layer = Input(30, name='albert_concat_layer', dtype=tf.float32)

  bert_q_layer = Input(21, name='bert_q_layer', dtype=tf.float32)
  bert_a_layer = Input(9, name='bert_a_layer', dtype=tf.float32)
  bert_qa_layer = Input(30, name='bert_qa_layer', dtype=tf.float32)
  bert_concat_layer = Input(30, name='bert_concat_layer', dtype=tf.float32)

  roberta_q_layer = Input(21, name='roberta_q_layer', dtype=tf.float32)
  roberta_a_layer = Input(9, name='roberta_a_layer', dtype=tf.float32)
  roberta_qa_layer = Input(30, name='roberta_qa_layer', dtype=tf.float32)
  roberta_concat_layer = Input(30, name='roberta_concat_layer', dtype=tf.float32)

  xlnet_q_layer = Input(21, name='xlnet_q_layer', dtype=tf.float32)
  xlnet_a_layer = Input(9, name='xlnet_a_layer', dtype=tf.float32)
  xlnet_qa_layer = Input(30, name='xlnet_qa_layer')
  xlnet_concat_layer = Input(30, name='xlnet_concat_layer', dtype=tf.float32)

  concat_layer = Concatenate()([albert_q_layer, albert_a_layer, albert_qa_layer, albert_concat_layer,
                                bert_q_layer, bert_a_layer, bert_qa_layer, bert_concat_layer,
                                roberta_q_layer, roberta_a_layer, roberta_qa_layer, roberta_concat_laye
r,
                                xlnet_q_layer, xlnet_a_layer, xlnet_concat_layer])

  print('concat_layer.shape:', concat_layer.shape)

  # x = Dense(512, activation='relu')(concat_layer)

  # x = Dense(128, activation='relu')(x)

  output = Dense(30, activation='sigmoid')(concat_layer)

  model = Model(inputs=[albert_q_layer, albert_a_layer, albert_qa_layer, albert_concat_layer,
                        bert_q_layer, bert_a_layer, bert_qa_layer, bert_concat_layer,
                        roberta_q_layer, roberta_a_layer, roberta_qa_layer, roberta_concat_layer,
```

```
                        xlnet_q_layer, xlnet_a_layer, xlnet_qa_layer, xlnet_concat_layer], outputs=ou
tput)
  return model
```

In [ ]:
```
model = create_model()
model.summary()
```

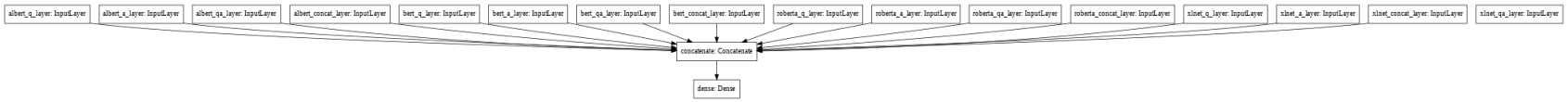
```
concat_layer.shape: (None, 330)
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
albert_q_layer (InputLayer)	[(None, 21)]	0	
albert_a_layer (InputLayer)	[(None, 9)]	0	
albert_qa_layer (InputLayer)	[(None, 30)]	0	
albert_concat_layer (InputLayer	[(None, 30)]	0	
bert_q_layer (InputLayer)	[(None, 21)]	0	
bert_a_layer (InputLayer)	[(None, 9)]	0	
bert_qa_layer (InputLayer)	[(None, 30)]	0	
bert_concat_layer (InputLayer)	[(None, 30)]	0	
roberta_q_layer (InputLayer)	[(None, 21)]	0	
roberta_a_layer (InputLayer)	[(None, 9)]	0	
roberta_qa_layer (InputLayer)	[(None, 30)]	0	
roberta_concat_layer (InputLaye	[(None, 30)]	0	
xlnet_q_layer (InputLayer)	[(None, 21)]	0	
xlnet_a_layer (InputLayer)	[(None, 9)]	0	
xlnet_concat_layer (InputLayer)	[(None, 30)]	0	
concatenate (Concatenate)	(None, 330)	0	albert_q_layer[0][0] albert_a_layer[0][0] albert_qa_layer[0][0] albert_concat_layer[0][0] bert_q_layer[0][0] bert_a_layer[0][0] bert_qa_layer[0][0] bert_concat_layer[0][0]

```
                                                                  roberta_q_layer[0][0]
                                                                  roberta_a_layer[0][0]
                                                                  roberta_qa_layer[0][0]
                                                                  roberta_concat_layer[0][0]
                                                                  xlnet_q_layer[0][0]
                                                                  xlnet_a_layer[0][0]
                                                                  xlnet_concat_layer[0][0]
__________________________________________________________________________________________________
xlnet_qa_layer (InputLayer)     [(None, 30)]         0
__________________________________________________________________________________________________
dense (Dense)                   (None, 30)           9930        concatenate[0][0]
==================================================================================================
Total params: 9,930
Trainable params: 9,930
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [ ]:
```
tf.keras.utils.plot_model(model, to_file='model.png',
                          show_shapes=False,
                          show_layer_names=True,
                          rankdir='TB',
                          expand_nested=False, dpi=48
                          )
```

Out[ ]:



In [ ]:
```
# Defining rhos metric
from scipy.stats import spearmanr
def compute_spearmanr_ignore_nan(trues, preds):
  rhos = []
  for tcol, pcol in zip(np.transpose(trues), np.transpose(preds)):
    rhos.append(spearmanr(tcol, pcol).correlation)
  return np.nanmean(rhos)
```

In [ ]:
```
# making rhos tensorflow graph compatible
def rhos(y, y_pred):
  return tf.py_function(compute_spearmanr_ignore_nan, (y, y_pred), tf.double)
metrics = [rhos]
```

In [ ]:
```
# Train data
train_data = {
    'albert_q_layer' : albert_train_q.values,
    'albert_a_layer' : albert_train_a.values,
    'albert_qa_layer' : albert_train_qa.values,
    'albert_concat_layer' : albert_train_concat.values,
    'bert_q_layer' : bert_train_q.values,
    'bert_a_layer' : bert_train_a.values,
    'bert_qa_layer' : bert_train_qa.values,
    'bert_concat_layer' : bert_train_concat.values,
    'roberta_q_layer' : roberta_train_q.values,
    'roberta_a_layer' : roberta_train_a.values,
    'roberta_qa_layer' : roberta_train_qa.values,
    'roberta_concat_layer' : roberta_train_concat.values,
    'xlnet_q_layer' : xlnet_train_q.values,
    'xlnet_a_layer' : xlnet_train_a.values,
    'xlnet_qa_layer' : xlnet_train_qa.values,
    'xlnet_concat_layer' : xlnet_train_concat.values
}
```

In [ ]:
```
# Test data
test_data = {
    'albert_q_layer' : albert_test_q.values,
    'albert_a_layer' : albert_test_a.values,
    'albert_qa_layer' : albert_test_qa.values,
    'albert_concat_layer' : albert_test_concat.values,
    'bert_q_layer' : bert_test_q.values,
    'bert_a_layer' : bert_test_a.values,
    'bert_qa_layer' : bert_test_qa.values,
    'bert_concat_layer' : bert_test_concat.values,
    'roberta_q_layer' : roberta_test_q.values,
    'roberta_a_layer' : roberta_test_a.values,
    'roberta_qa_layer' : roberta_test_qa.values,
    'roberta_concat_layer' : roberta_test_concat.values,
    'xlnet_q_layer' : xlnet_test_q.values,
    'xlnet_a_layer' : xlnet_test_a.values,
    'xlnet_qa_layer' : xlnet_test_qa.values,
    'xlnet_concat_layer' : xlnet_test_concat.values
}
```

In [ ]:
```
# Target values
target_label = X.values
```

Let's try with Binary Crossentropy as loss function

In [ ]:
```
# Compiling and training the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=metrics)
model.fit(train_data, target_label, epochs=5, batch_size=4)
```

```
Epoch 1/5
   7/1520 [..............................] - ETA: 25s - loss: 0.7367 - rhos: -0.0030

/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2534: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[:, None]
/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2535: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[None, :]
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in greater
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in less
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:1912: RuntimeWarning: in
valid value encountered in less_equal
  cond2 = cond0 & (x <= _a)

1520/1520 [==============================] - 29s 19ms/step - loss: 0.4160 - rhos: 0.2818
Epoch 2/5
1520/1520 [==============================] - 30s 19ms/step - loss: 0.3679 - rhos: 0.4385
Epoch 3/5
1520/1520 [==============================] - 30s 20ms/step - loss: 0.3530 - rhos: 0.4887
Epoch 4/5
1520/1520 [==============================] - 30s 20ms/step - loss: 0.3441 - rhos: 0.5102
Epoch 5/5
1520/1520 [==============================] - 30s 20ms/step - loss: 0.3383 - rhos: 0.5254
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f8c4e7a88d0>
```

In [ ]:
```
pred = model.predict(test_data)
df = pd.concat([test['qa_id'], pd.DataFrame(pred, columns=submission.columns[1:])], axis=1)
df.to_csv('output_bce.csv', index=False)
```

In [ ]:
```

```

Let's try with Mean squared error as loss function

In [ ]:
```
model = create_model()
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(loss='mean_squared_error', optimizer=optimizer, metrics=metrics)
model.fit(train_data, target_label, epochs=20, batch_size=32, validation_split=0.15)
```

```
concat_layer.shape: (None, 330)
Epoch 1/20
  7/162 [>.............................] - ETA: 3s - loss: 0.1781 - rhos: 0.0372

/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2534: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[:, None]
/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2535: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[None, :]
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in greater
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in less
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:1912: RuntimeWarning: in
valid value encountered in less_equal
  cond2 = cond0 & (x <= _a)
```

```
162/162 [==============================] - 5s 33ms/step - loss: 0.0710 - rhos: 0.2189 - val_loss: 0.
0449 - val_rhos: 0.2965
Epoch 2/20
162/162 [==============================] - 5s 31ms/step - loss: 0.0389 - rhos: 0.3613 - val_loss: 0.
0363 - val_rhos: 0.3751
Epoch 3/20
162/162 [==============================] - 5s 32ms/step - loss: 0.0330 - rhos: 0.4170 - val_loss: 0.
0330 - val_rhos: 0.4077
Epoch 4/20
162/162 [==============================] - 5s 31ms/step - loss: 0.0301 - rhos: 0.4468 - val_loss: 0.
0315 - val_rhos: 0.4289
Epoch 5/20
162/162 [==============================] - 5s 32ms/step - loss: 0.0283 - rhos: 0.4650 - val_loss: 0.
0301 - val_rhos: 0.4444
Epoch 6/20
162/162 [==============================] - 5s 30ms/step - loss: 0.0270 - rhos: 0.4781 - val_loss: 0.
0290 - val_rhos: 0.4549
Epoch 7/20
162/162 [==============================] - 5s 30ms/step - loss: 0.0259 - rhos: 0.4923 - val_loss: 0.
0283 - val_rhos: 0.4668
Epoch 8/20
162/162 [==============================] - 5s 32ms/step - loss: 0.0251 - rhos: 0.5037 - val_loss: 0.
0277 - val_rhos: 0.4753
Epoch 9/20
162/162 [==============================] - 5s 33ms/step - loss: 0.0243 - rhos: 0.5136 - val_loss: 0.
0273 - val_rhos: 0.4824
Epoch 10/20
162/162 [==============================] - 5s 33ms/step - loss: 0.0238 - rhos: 0.5205 - val_loss: 0.
0269 - val_rhos: 0.4889
Epoch 11/20
162/162 [==============================] - 5s 31ms/step - loss: 0.0233 - rhos: 0.5268 - val_loss: 0.
0265 - val_rhos: 0.4952
Epoch 12/20
162/162 [==============================] - 5s 32ms/step - loss: 0.0229 - rhos: 0.5312 - val_loss: 0.
0264 - val_rhos: 0.5000
Epoch 13/20
162/162 [==============================] - 5s 32ms/step - loss: 0.0224 - rhos: 0.5341 - val_loss: 0.
0259 - val_rhos: 0.5042
Epoch 14/20
162/162 [==============================] - 5s 32ms/step - loss: 0.0222 - rhos: 0.5422 - val_loss: 0.
0257 - val_rhos: 0.5091
Epoch 15/20
162/162 [==============================] - 6s 35ms/step - loss: 0.0218 - rhos: 0.5452 - val_loss: 0.
```

```
0256 - val_rhos: 0.5118
Epoch 16/20
162/162 [==============================] - 5s 33ms/step - loss: 0.0216 - rhos: 0.5497 - val_loss: 0.
0254 - val_rhos: 0.5161
Epoch 17/20
162/162 [==============================] - 5s 29ms/step - loss: 0.0213 - rhos: 0.5526 - val_loss: 0.
0250 - val_rhos: 0.5189
Epoch 18/20
162/162 [==============================] - 5s 31ms/step - loss: 0.0210 - rhos: 0.5557 - val_loss: 0.
0248 - val_rhos: 0.5218
Epoch 19/20
162/162 [==============================] - 5s 31ms/step - loss: 0.0208 - rhos: 0.5588 - val_loss: 0.
0249 - val_rhos: 0.5243
Epoch 20/20
162/162 [==============================] - 5s 31ms/step - loss: 0.0207 - rhos: 0.5615 - val_loss: 0.
0247 - val_rhos: 0.5265
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f65eefca3c8>
```

In [ ]:
```
test = pd.read_csv('test.csv')
submission = pd.read_csv('sample_submission.csv')

pred = model.predict(test_data)
df = pd.concat([test['qa_id'], pd.DataFrame(pred, columns=submission.columns[1:])], axis=1)
df.to_csv('output_mse.csv', index=False)
df.head(5)
```

Out[ ]:

	qa_id	question_asker_intent_understanding	question_body_critical	question_conversational	question_expect_short_answer	question_fa
0	39	0.929953	0.588778	0.173808	0.704667	
1	46	0.873381	0.401949	0.010194	0.762220	
2	70	0.888685	0.683490	0.021943	0.855611	
3	132	0.844835	0.451844	0.013114	0.745299	
4	200	0.908349	0.335955	0.035499	0.765231	

In [ ]:

Let's try Mean of predicted values from all the previous transformr models

In [ ]:
```
# weights for albert, bert, roberta and xlnet predictions.
a1, a2 = 0, 0
b1, b2 = 1, 1
r1, r2 = 1, 1
x1, x2 = 0, 1

pred_mean = ((albert_test_qa * a1 + albert_test_concat * a2)
             + (bert_test_qa * b1 + bert_test_concat * b2)
             + (roberta_test_qa * r1 + roberta_test_concat * r2)
             + (xlnet_test_qa * x1 + xlnet_test_concat * x2)) / (a1 + a2 + b1 + b2 + r1 + r2 + x1 + x2
)

# saving the predicted labels as dataframes
df = pd.concat([test['qa_id'], pd.DataFrame(optimize_ranks(pred_mean.values), columns=submission.colu
mns[1:])], axis=1)
df.to_csv('output.csv', index=False)b
```

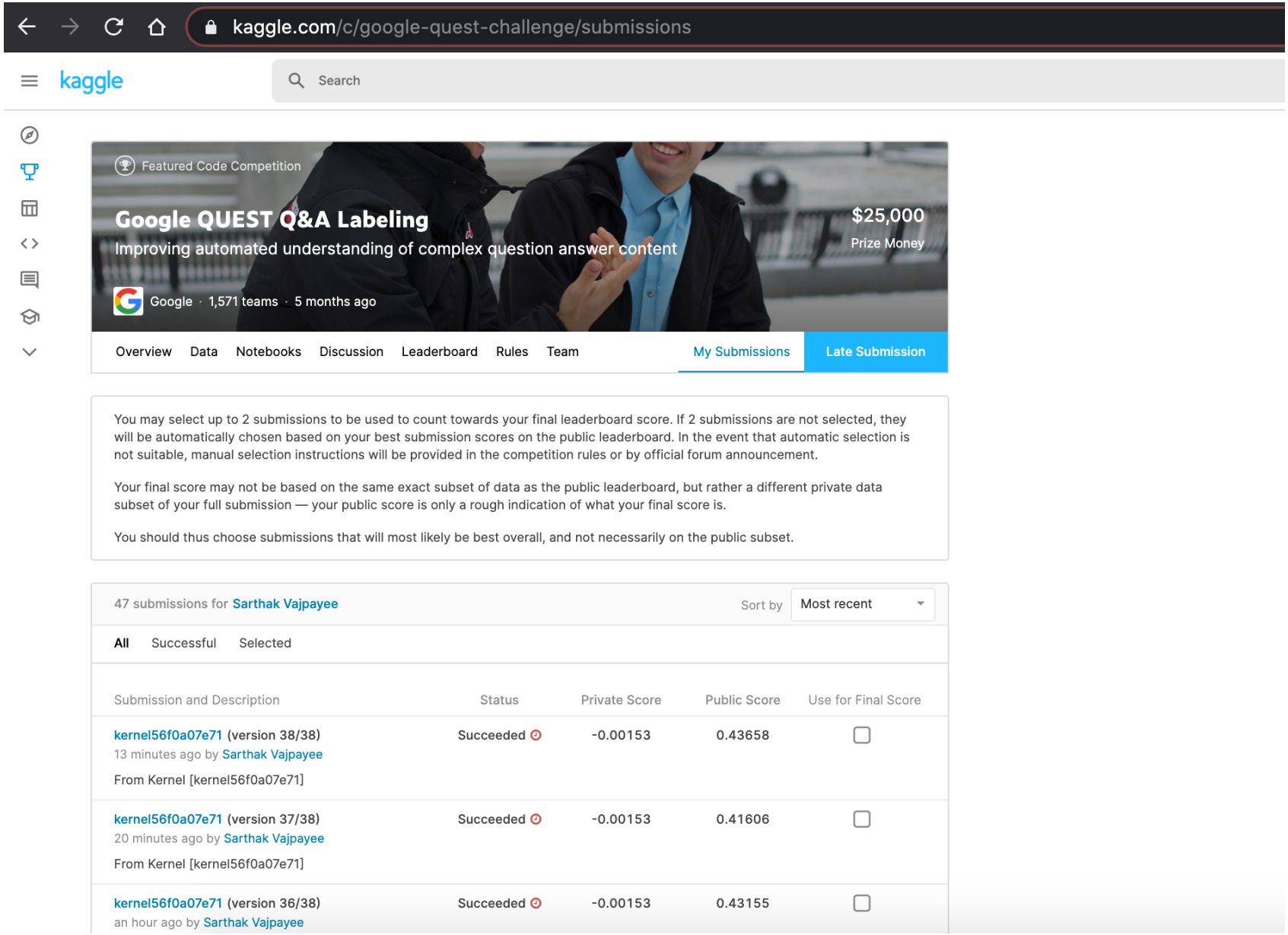
In [ ]:
```
df.head()
```

Out[ ]:

	qa_id	question_asker_intent_understanding	question_body_critical	question_conversational	question_expect_short_answer	question_fa
0	39	0.973333	0.733333	0.16	0.551111	
1	46	0.866667	0.520000	0.08	0.768889	
2	70	0.913333	0.733333	0.08	0.840000	
3	132	0.913333	0.533333	0.08	0.720000	
4	200	0.946667	0.506667	0.08	0.782222	

In [ ]:
```
%matplotlib inline
from IPython.display import Image
Image('score.png', width=1000, height=740)
```

Out[ ]:



In [ ]:
```
```

Using the above architecture, I was able to achieve a score of 0.43658 (top 4.4% in the kaggle leaderboard).

In [ ]: