CS640 Artificial Intelligence

Final Project

By Nicholas Zotalis, Aditya Agrawal,

Finn Jensen, Minh Le Nguyen, and Keith Tyser

Topic: Demographic Age and Race prediction of Twitter users

Table of Contents

I. Github Link

https://github.com/MinhNguyen99AI/CS640

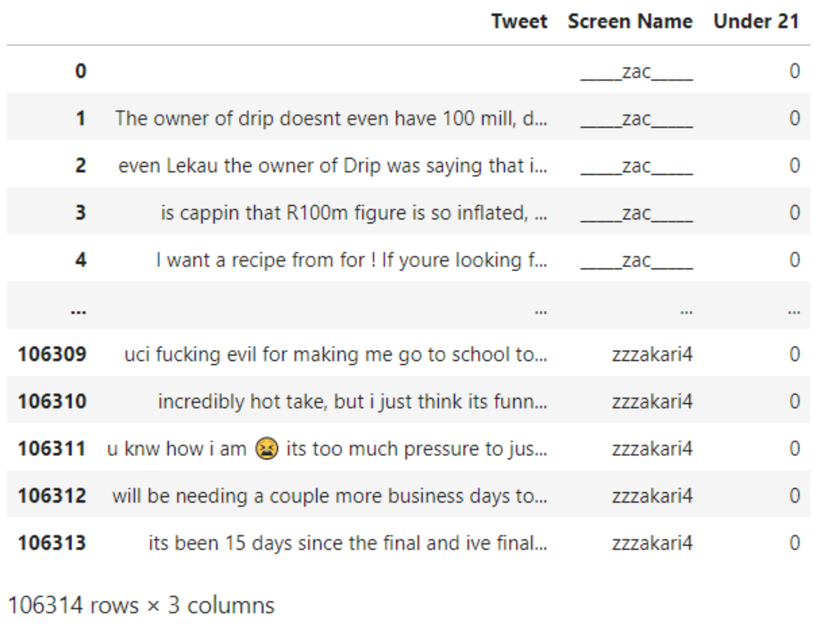
The important code and data is described in the readme, but we decided to leave everything that we did in the repository, even the failed attempts, in order to show all our work.

II. Preprocessing Data

We were given different datasets for age and race predictions. In both cases, since the goal was to predict the age/race of a **user** given their tweets, we reorganized each dataset such that each entry contained a user, their label, and all their tweets concatenated together. In both datasets, the tweets were run through preprocessing to remove links, emojis, etc.

Age Dataset

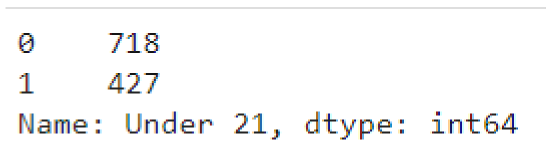
All tweets were extracted from tweets.json using the screen name extracted from labeled_users.csv. The necessary columns were extracted from labeled_users and a dataset was created with tweets, screen names, and the label for "Under 21". The tweets were preprocessed to remove @Usernames, #Hashtags, HTTPS links, HTML Tags, and punctuations. This is the main dataset for the age prediction models



	Tweet	Screen Name	Under 21
0		____zac____	0
1	The owner of drip doesnt even have 100 mill, d...	____zac____	0
2	even Lekau the owner of Drip was saying that i...	____zac____	0
3	is cappin that R100m figure is so inflated, ...	____zac____	0
4	I want a recipe from for ! If youre looking f...	____zac____	0
...	...	...	...
106309	uci fucking evil for making me go to school to...	zzzakari4	0
106310	incredibly hot take, but i just think its funn...	zzzakari4	0
106311	u knw how i am 😫 its too much pressure to jus...	zzzakari4	0
106312	will be needing a couple more business days to...	zzzakari4	0
106313	its been 15 days since the final and ive final...	zzzakari4	0

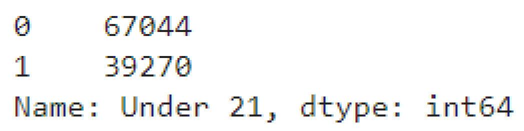
106314 rows × 3 columns

The dataset has 1145 unique screen names, with approximately 100 tweets per user. The division of labels by users is:-



```
0    718
1    427
Name: Under 21, dtype: int64
```

The Division of labels by each tweets is:-

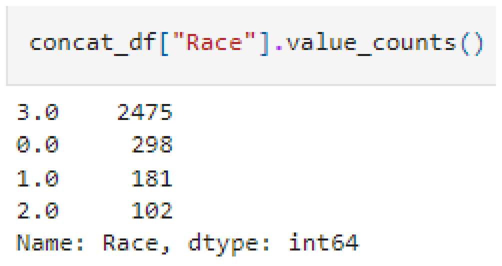


```
0    67044
1    39270
Name: Under 21, dtype: int64
```

The labels are unbalanced and this causes problems with some models. Oversampling was used to try to deal with this problem, as detailed later.

Race Dataset

The race dataset contained labels 1, 2, 3, 4, 5. The 5 label was dropped since we were not required to predict the multiracial class. The rest of the labels were lowered by 1 to fit in with a neural network's multi-class output. The final race dataset used for training had the following distribution of labels:



```
concat_df["Race"].value_counts()
```

```
3.0    2475
0.0     298
1.0     181
2.0     102
Name: Race, dtype: int64
```

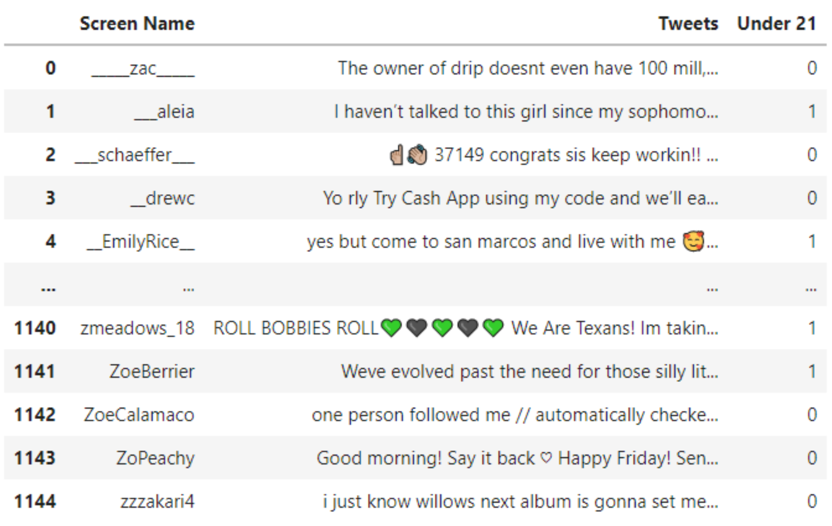
We have 3056 training samples, with some 81% belonging to class 3 (white). Our attempts to mitigate this imbalance in the data are explained later in the report.

III. Building Models and Results

1. Age Prediction Models

a. Naive Bayes Classifier

For this model, the main dataset was modified. A new Dataframe was created where all the tweets for each user were concatenated into a single string. This created a large string of tweets for each user. The distribution of screen names is the same as the main dataset.



	Screen Name	Tweets	Under 21
0	____zac____	The owner of drip doesnt even have 100 mill,...	0
1	__aleia	I haven't talked to this girl since my sophomo...	1
2	__schaeffer__	👍👏 37149 congrats sis keep workin!! ...	0
3	__drewc	Yo rly Try Cash App using my code and we'll ea...	0
4	__EmilyRice__	yes but come to san marcos and live with me 🥰...	1
...	...	...	...
1140	zmeadows_18	ROLL BOBBIES ROLL💚🖤💚🖤💚 We Are Texans! Im takin...	1
1141	ZoeBerrier	Weve evolved past the need for those silly lit...	1
1142	ZoeCalamaco	one person followed me // automatically checke...	0
1143	ZoPeachy	Good morning! Say it back ♡ Happy Friday! Sen...	0
1144	zzzakari4	i just know willows next album is gonna set me...	0

To extract features from the tweets, sklearn's CountVectorizer() was used, with the option to remove English stop words enabled.

```
stop_words_vectorizer=CountVectorizer(stop_words='english')
```

Next the Naive Bayes Classifier was implemented with sklearn's MultinomialNB() function.

```
nb = MultinomialNB()
```

Next a 5-fold cross validation split was done on the dataset. The training and testing was done using these splits. The average result of the 5-fold cross validation for this model is:-

```
K-fold 1/5
--------------------
Train accuracy- 0.987
Test accuracy- 0.646
K-fold 2/5
--------------------
Train accuracy- 0.992
Test accuracy- 0.716
K-fold 3/5
--------------------
Train accuracy- 0.988
Test accuracy- 0.703
K-fold 4/5
--------------------
Train accuracy- 0.991
Test accuracy- 0.62
K-fold 5/5
--------------------
Train accuracy- 0.989
Test accuracy- 0.655
```

```
Overall Accuracy- 0.668

------(Age >= 21)------

Precision- 0.67
Recall- 0.932
F1- 0.779

------(Age < 21)------

Precision- 0.659
Recall- 0.228
F1- 0.335
```

```
Confusion Matrix-
 [[669.  49.]
 [331.  96.]]
```

The results show an accuracy of 66.8 % and a poor recall for the label 1 (Age<21). This can be attributed to the imbalance in labels in the dataset (718 for label 0 and 427 for label 1). This performance is not satisfactory as it mostly classifies the tweets as label 0 so we tried more models.

b. Logistic Regression

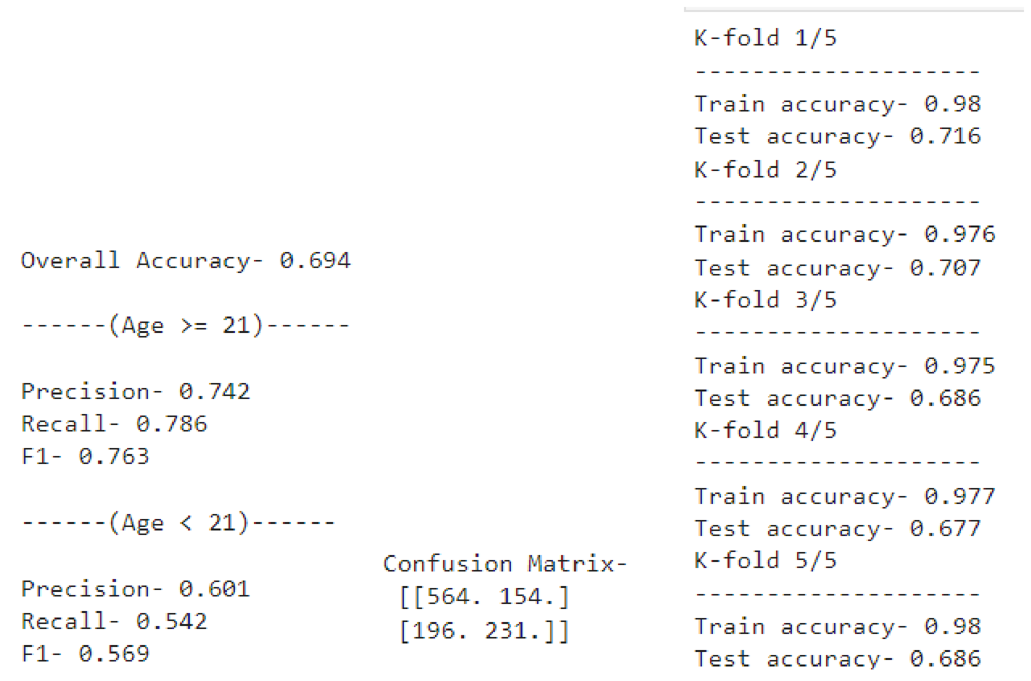
This model uses the same dataframe of concatenated tweets used in the Naive Bayes Classifier shown above. To extract the features from the tweets, sklearn's TfidfVectorizer() was used, with the option to remove English stop words enabled.

```
tfidf = TfidfVectorizer(stop_words='english')
```

The logistic regression classifier was implemented with Sklearn's LogisticRegression() method with class weight balanced.

```
log = LogisticRegression(class_weight = 'balanced')
```

Next a 5-fold cross validation split was done like above The average result of the 5-fold cross validation for this model is:-



```
Overall Accuracy- 0.694

------(Age >= 21)------

Precision- 0.742
Recall- 0.786
F1- 0.763

------(Age < 21)------

Precision- 0.601
Recall- 0.542
F1- 0.569
```

```
Confusion Matrix-
 [[564. 154.]
 [196. 231.]]
```

```
K-fold 1/5
--------------------
Train accuracy- 0.98
Test accuracy- 0.716
K-fold 2/5
--------------------
Train accuracy- 0.976
Test accuracy- 0.707
K-fold 3/5
--------------------
Train accuracy- 0.975
Test accuracy- 0.686
K-fold 4/5
--------------------
Train accuracy- 0.977
Test accuracy- 0.677
K-fold 5/5
--------------------
Train accuracy- 0.98
Test accuracy- 0.686
```

The accuracy has improved by 2.6 % compared to the Naive Bayes Classifier. We also got a way better recall for label 1 (Age<21). The recall for label 0 (Age>=21) dropped. Hence this model deals with the class imbalance better than the Naive Bayes Classifier, and predicts label 1 better and gives us closer recall for both labels.

c. BERT Model Classification

A BERT Model Classifier is created using Pytorch Neural Networks. This is chosen because BERT is very useful for text classification and other NLP applications.

The first layer of our model is the BERT Model with “bert-base-cased” pretrained weight. The second layer is the Dropout Layer to control regularization. The third layer is the Linear Layer with output size 2, denoting the number of classes. (because it is a requirement by the CrossEntropyLoss used).

```
class Classifier(torch.nn.Module):
    def __init__(self):

        super(Classifier, self).__init__()

        self.bert_model=BertModel.from_pretrained("bert-base-cased")

        self.dropout = nn.Dropout(p=0.3)

        self.linear = nn.Linear(self.bert_model.config.hidden_size,2)

    def forward(self,input_ids, attention_mask):

        last_hidden_layer,pooled_output = self.bert_model(input_ids=input_ids,attention_mask=attention_mask, return_dict=False)

        dropout_output = self.dropout(pooled_output)

        linear_output = self.linear(dropout_output)

        return linear_output
```

For extracting features from the tweets, BERT Tokenizer is used with “bert-base-cased” pretrained weights. The encoder in this tokenizer returns input_ids and attention mask for each tweet.

```
tokenizer = BertTokenizer.from_pretrained("bert-base-cased")
```

```
encoding_input = self.tokenizer.encode_plus(tweet,max_length=self.max_len, add_special_tokens=True,\
                                            return_token_type_ids=False,pad_to_max_length=True, return_attention_mask=True,\
                                            return_tensors='pt',truncation=True)
```

A Pytorch Dataloader is used that returns the tweet, label, input_ids and attention mask for each tweet. The max length for encoding is set to 160, since the longest tweet in the dataset is 150 tokens long.

For training, the loss is CrossEntropyLoss, the optimizer is Adam, the learning rate is 3.1e-5, the number of epochs is 4 and the batch size is 64. A linear scheduler is also used to decrease the learning rate during training.

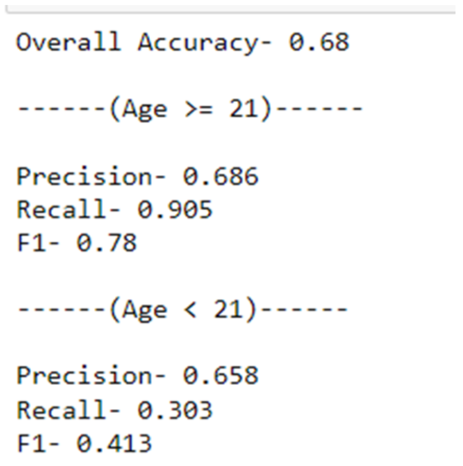
A 5-fold cross validation split is done on the main dataset (no concatenation) using the screen_names. The creation of train and test dataset using screen name list can be seen in cell 41 of agePrediction.ipynb.

```
screen_names_list = df['Screen Name'].unique()

kf = KFold(n_splits=5, shuffle = True, random_state=24)
```

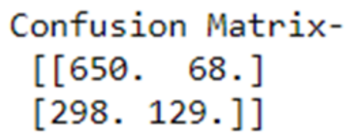
```
for train_tweets_sn, test_tweets_sn in kf.split(screen_names_list):
```

During training, the data loader is loaded with all tweets in the training dataset regardless of users and the BERT Model is trained using all these tweets.

During testing, the data loader is loaded with tweets for a particular user, the model is tested, the prediction for each tweet is taken, the most frequent prediction is the final prediction for that user and this process is repeated for all users in the test screen_names list.
There are 4 epochs in each split of the 5-fold cross validation split. The results of this are:-



```
Overall Accuracy- 0.68

------(Age >= 21)------

Precision- 0.686
Recall- 0.905
F1- 0.78

------(Age < 21)------

Precision- 0.658
Recall- 0.303
F1- 0.413
```

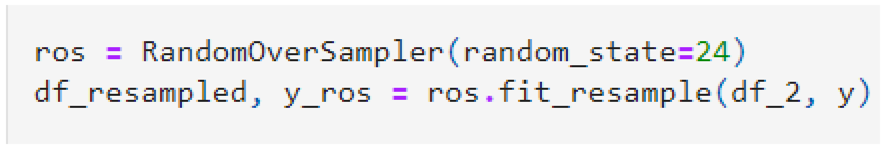


```
Confusion Matrix-
 [[650.  68.]
 [298. 129.]]
```

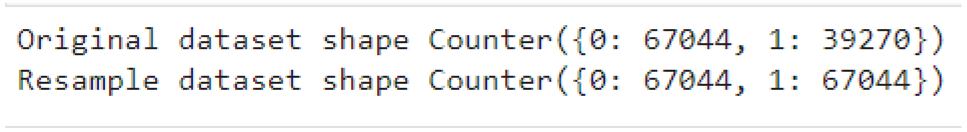
The accuracy of this model is 68%. It is better than our Naive Bayes but not as good as the Logistic Regression. The recall for label 1 is poor in this model as well. This is due to the same reason as before, imbalance in the label distribution. (67044 tweets for label 0, 39270 tweets for label 1). We next tried to improve this accuracy for the same model.

d. <u>BERT Model Classification with Oversampling</u>.

We used the *exact same BERT Model* described above but changed the dataset given to it. We tried Oversampling on our dataset to make the distribution of labels equal.

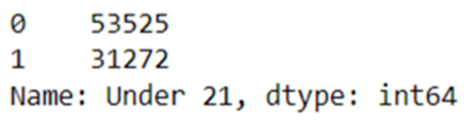


```
ros = RandomOverSampler(random_state=24)
df_resampled, y_ros = ros.fit_resample(df_2, y)
```

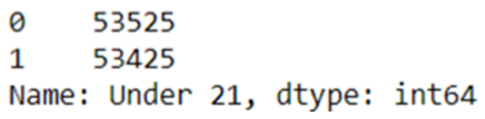


```
Original dataset shape Counter({0: 67044, 1: 39270})
Resample dataset shape Counter({0: 67044, 1: 67044})
```

We managed to do so. The distribution of labels by user still stays the same as before (718/427). But now during the training of our BERT Model, the distribution of labels is equal. Below is that distribution before and after oversampling.

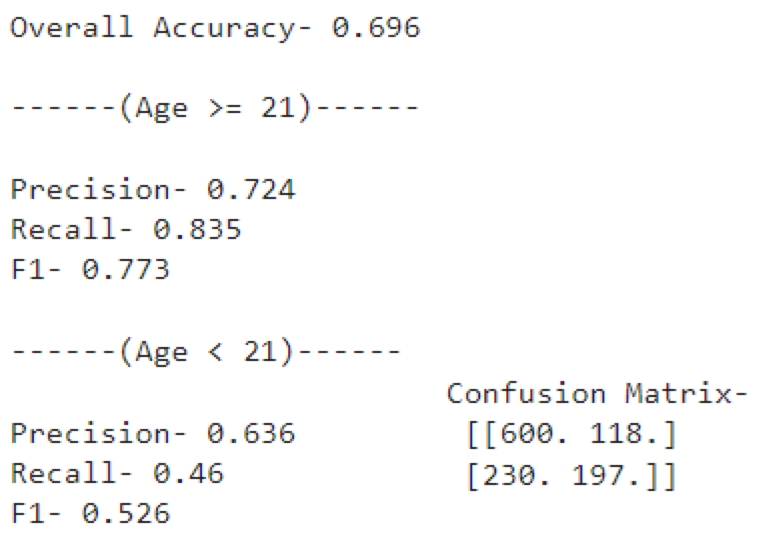


```
0    53525
1    31272
Name: Under 21, dtype: int64
```



```
0    53525
1    53425
Name: Under 21, dtype: int64
```

Using the same method for 5-fold cross validation as the BERT Model above, the following results were obtained.



```
Overall Accuracy- 0.696

------(Age >= 21)------

Precision- 0.724
Recall- 0.835
F1- 0.773

------(Age < 21)------
                                Confusion Matrix-
Precision- 0.636                 [[600. 118.]
Recall- 0.46                      [230. 197.]]
F1- 0.526
```

The accuracy with this method is 69.6%. The accuracy is 1.6 % better than without oversampling. The recall has also increased compared to before. So bringing balance to the dataset does help a little with getting more predictions for label 1. The recall is still not as good as Logistic Regression.

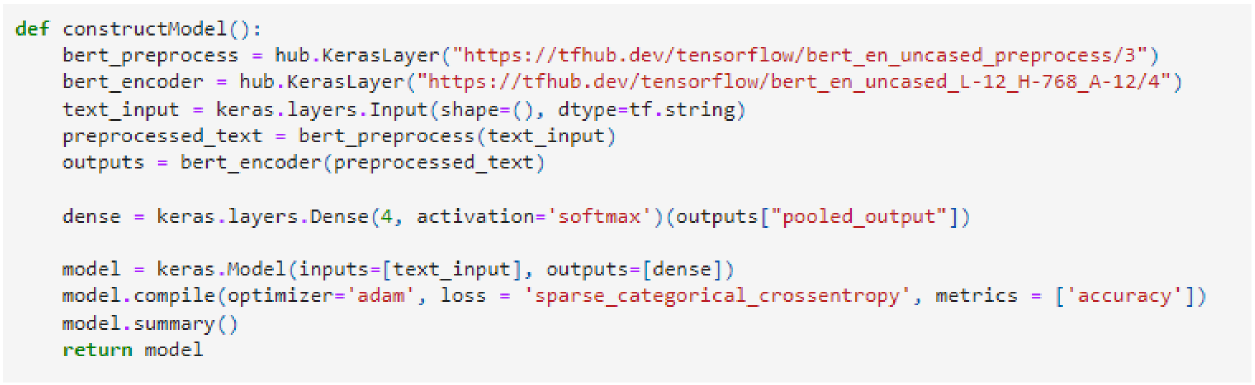
e. Conclusion for Age Prediction

Out of our four methods, Logistic Regression and BERT Model with Oversampling have the best 5-fold average accuracy of approximately 69.5 %. The best recall for label 1 (Age<21) is given by Logistic Regression so that model is the best model for handling the imbalanced data and best for this problem of age prediction.

2. Race Prediction Model with Tweets

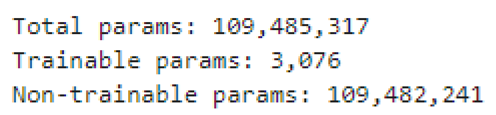
a. First Attempt: BERT classification + Keras Neural Network

Models Decision: Since the biggest challenge in any NLP problem is vectorizing the inputs, we decided to see if BERT's pre-trained tokenizer could help us out.



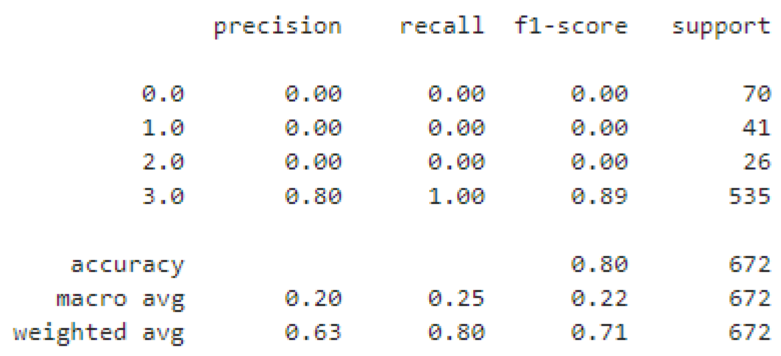
```python
def constructModel():
    bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
    bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
    text_input = keras.layers.Input(shape=(), dtype=tf.string)
    preprocessed_text = bert_preprocess(text_input)
    outputs = bert_encoder(preprocessed_text)

    dense = keras.layers.Dense(4, activation='softmax')(outputs["pooled_output"])

    model = keras.Model(inputs=[text_input], outputs=[dense])
    model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
    model.summary()
    return model
```

The model takes BERT's tokenization of the concatenated tweets for each user and connects them to a normal output layer with a softmax activation function.



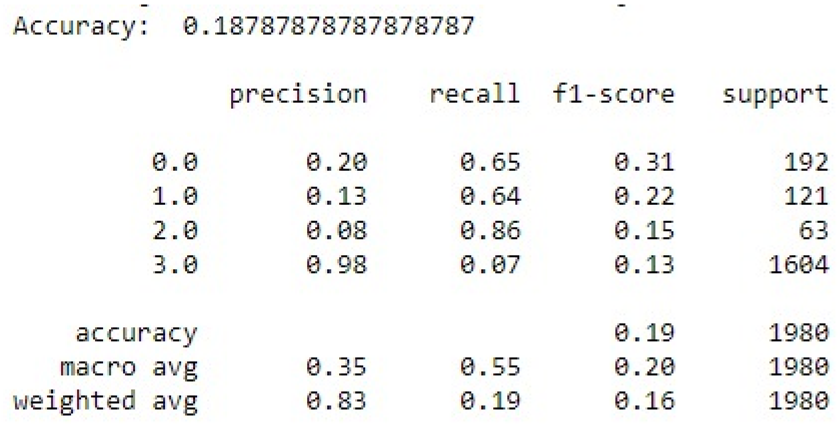
```
Total params: 109,485,317
Trainable params: 3,076
Non-trainable params: 109,482,241
```

These are the test results after training the model for several epochs:



```
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        70
         1.0       0.00      0.00      0.00        41
         2.0       0.00      0.00      0.00        26
         3.0       0.80      1.00      0.89       535

    accuracy                           0.80       672
   macro avg       0.20      0.25      0.22       672
weighted avg       0.63      0.80      0.71       672
```

The model has learned to predict 3 for every example. By doing this it is achieving 80% accuracy, but this approach is useless because it cannot distinguish any of the other classes. We decided that this model is a failure.
To fix this label imbalance, we tried oversampling the minority class to make all of the class labels equal.



```
Accuracy:  0.18787878787878787

              precision    recall  f1-score   support

         0.0       0.20      0.65      0.31       192
         1.0       0.13      0.64      0.22       121
         2.0       0.08      0.86      0.15        63
         3.0       0.98      0.07      0.13      1604

    accuracy                           0.19      1980
   macro avg       0.35      0.55      0.20      1980
weighted avg       0.83      0.19      0.16      1980
```

After retraining our model with the new resampled dataset, we achieved these results. The recall and precision both improved which was good, however the accuracy and f1-score suffered greatly. It was good to see the model predicting all of the class labels, but this oversampling technique did not work to improve our model.
Oversampling the minority class likely caused overfitting since it is making exact copies of the minority class examples. This is probably fooling the model into thinking it is making accurate predictions for many examples when in reality it is just for one replicated example.

b. Second Attempt: TF-IDF + Logistic regression

```
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
log = LogisticRegression(class_weight = 'balanced', max_iter = 10 ** 6)
```

Models Decision: After our neural network model didn't return great results, we decided to try a simple logistic regression model. We used balanced class weights to address the class imbalance issue. The number of iterations was capped at an arbitrarily high value since the default value was not enough for convergence to occur.

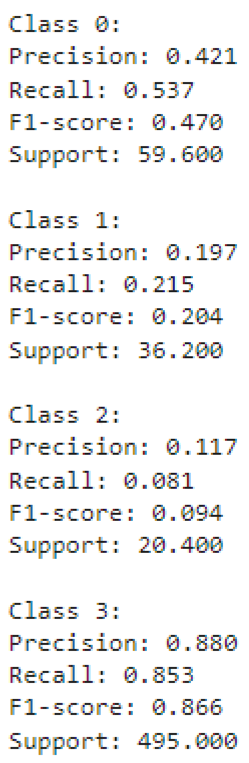
We used TfidfVectorizer on the tweets which used the Term Frequency - Inverse Document Frequency technique to vectorize the tweets and turn them into features before training our model. Through trial and error we settled on 5000 features, and left the ngram range to the sklearn default, which is unigrams. We experimented all the way up to 5-grams but achieved the best results with simple unigrams.

```
Fold 5:

Accuracy: 0.764

              precision    recall  f1-score   support

         0.0       0.37      0.46      0.41        50
         1.0       0.19      0.17      0.18        40
         2.0       0.00      0.00      0.00        17
         3.0       0.88      0.87      0.87       504

    accuracy                           0.76       611
   macro avg       0.36      0.38      0.37       611
weighted avg       0.77      0.76      0.77       611


[[0.46       0.06       0.04       0.44      ]
 [0.175      0.175      0.05       0.6       ]
 [0.11764706 0.17647059 0.         0.70588235]
 [0.06150794 0.04761905 0.02380952 0.86706349]]
```

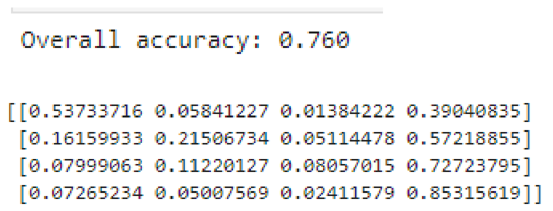
With 5-fold cross validation, the logistic regression model returned really good results. The 76% accuracy was slightly lower than our initial neural network model accuracy of 80%, but the 77 f1-score was better than the 71 f1-score of the neural network model. The recall and precision of this model was also better than the recall and precision for the neural network.

Since this model improved many metrics from the neural network model, we decided to choose this as our final model.

c. <u>K-Fold Validation Testing for Race Prediction Models:</u>



```
Class 0:
Precision: 0.421
Recall: 0.537
F1-score: 0.470
Support: 59.600

Class 1:
Precision: 0.197
Recall: 0.215
F1-score: 0.204
Support: 36.200

Class 2:
Precision: 0.117
Recall: 0.081
F1-score: 0.094
Support: 20.400

Class 3:
Precision: 0.880
Recall: 0.853
F1-score: 0.866
Support: 495.000
```



```
Overall accuracy: 0.760

[[0.53733716 0.05841227 0.01384222 0.39040835]
 [0.16159933 0.21506734 0.05114478 0.57218855]
 [0.07999063 0.11220127 0.08057015 0.72723795]
 [0.07265234 0.05007569 0.02411579 0.85315619]]
```

The overall accuracy of 76%, while lower than the neural network's 80%, is much better since this model is not just predicting 3 for every class. Precision and recall are reasonable for class 0 and non-zero for class 1 and class 2, which are huge improvements over the neural network's performance.

3. Race Prediction Model with Profile Pictures

We attempted to build a neural network to classify race based on profile pictures for which we had the labels.

```
model = keras.Sequential()

model.add(keras.layers.Conv2D(filters=3, kernel_size=(3,3), activation='relu', input_shape=(224,224,3)))
model.add(keras.layers.Dropout(.25))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(20, activation='relu'))
model.add(keras.layers.Dense(4, activation='softmax'))
model.summary()
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 3)       84

 dropout (Dropout)           (None, 222, 222, 3)       0

 flatten (Flatten)           (None, 147852)            0

 dense (Dense)               (None, 20)                2957060

 dense_1 (Dense)             (None, 4)                 84

=================================================================
Total params: 2,957,228
Trainable params: 2,957,228
Non-trainable params: 0
```

Other models achieved extremely high training accuracy but overfit the training set. This model was less overfit.

```
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        70
         1.0       0.00      0.00      0.00        41
         2.0       0.00      0.00      0.00        26
         3.0       0.80      1.00      0.89       535

    accuracy                           0.80       672
   macro avg       0.20      0.25      0.22       672
weighted avg       0.63      0.80      0.71       672
```

This model ran into the same problem as the race prediction model using tweets. It learned to only predict the 3 class label. That is not a good model and would not generalize well to other datasets. Therefore, because of this and the problem that so many profile pictures are cartoons/animated images, we decided profile pictures would not be useful for this project.

III. Conclusion and Insights

For both age and race classification, the most effective methods we tried involved logistic regression. This is likely due to the imbalanced labels. Our test accuracy for race classification was 76%, which is an improvement on the baseline model accuracy of 67%. Our model was similar to the baseline model, but we believe that concatenating tweets by user was a key insight. Our best testing accuracy for age classification was 69.5% which may or may not exceed a baseline model, since the given baseline was based on a different age training set.

IV. Stats Age and Race

Race Prediction Attempt 1a

- Model: BERT classification with Keras Neural Network
- Features: Concatenated tweets
- Data: Dataframe with concatenated tweets of every user.
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

Accuracy: **80%**

Performance	African - American (0)	Latino - Hispanic (1)	Asian (2)	White (3)
Precision	0%	0%	0%	80%
Recall	0%	0%	0%	100%
F1	0%	0%	0%	89%

Race Prediction Attempt 1b

- Model: BERT classification, Keras Neural Network and RandomOverSampler
- Features: Concatenated tweets
- Data: Dataframe with concatenated tweets of every user. Minority classes are oversampled and class labels are equal.
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

Accuracy: **18.78%**

Performance	African - American (0)	Latino - Hispanic (1)	Asian (2)	White (3)
Precision	20%	13%	8%	98%
Recall	65%	64%	86%	7%
F1	31%	22%	15%	13%

Race Prediction Attempt 2

- Model: Logistic Regression + TF-IDF
- Features: TF-IDF on concatenated tweets
- Data: New Dataframe with Concatenated Tweets for every user
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

Accuracy: **76.0%**

Confusion Matrix	African - American (0)	Latino - Hispanic (1)	Asian (2)	White (3)
African - American (0)	.537	.058	.014	.390
Latino - Hispanic (1)	.161	.215	.051	.572
Asian (2)	.079	.112	.081	.727
White (3)	.073	.050	.024	.853

Performance	African - American (0)	Latino - Hispanic (1)	Asian (2)	White (3)
Precision	37%	19%	0%	88%
Recall	46%	17%	0%	87%
F1	41%	18%	0%	87%

Age Prediction Attempt 1

- Model: Naive Bayes Classifier.
- Features: CountVectorizer on concatenated tweets with stop words removed.
- Data: New Dataframe with Concatenated Tweets for every user
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

Accuracy: **66.8%**

Confusion Matrix	<21	>=21
<21	96	331
>=21	49	669

Performance	<21	>=21
Precision	65.9%	67%
Recall	22.8%	93.2%
F1	33.5%	77.9%

Age Prediction Attempt 2

- Model: Logistic Regression.
- Features: TF-IDF Vectorizer on concatenated tweets with stop words removed.
- Data: New Dataframe with Concatenated Tweets for every user
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

Accuracy: **69.4%**

Confusion Matrix	<21	>=21
<21	231	196
>=21	154	564

Performance	<21	>=21
Precision	60.1 %	74.2 %
Recall	54.2 %	78.6 %
F1	56.9 %	76.3 %

<u>**Age Prediction Attempt 3**</u>

- Model: <u>BERT Model</u> with “bert-based-cased” pretrained weights
- Features: <u>BERT Tokenizer</u> with “bert-based-cased” pretrained weights on every tweet.
- Data: The dataset is used as initially created. No concatenation. <u>No Oversampling.</u>
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

<u>Accuracy: **68%**</u>

Confusion Matrix	<21	>=21
<21	129	298
>=21	68	650

Performance	<21	>=21
Precision	65.8 %	68.6 %
Recall	30.3 %	90.5 %
F1	41.3 %	78 %

Age Prediction Attempt 4

- Model: BERT Model with “bert-based-cased” pretrained weights
- Features: BERT Tokenizer with “bert-based-cased” pretrained weights on every tweet.
- Data: Oversampling is done on the dataset initially created to balance labels. No concatenation.
- Metric: 5-fold cross validation average for every performance metrics. 5-fold split done by screen_name (user).

Accuracy: **69.6%**

Confusion Matrix	<21	>=21
<21	197	230
>=21	118	600

Performance	<21	>=21
Precision	63.6 %	72.4 %
Recall	46 %	83.5 %
F1	52.6 %	77.3 %