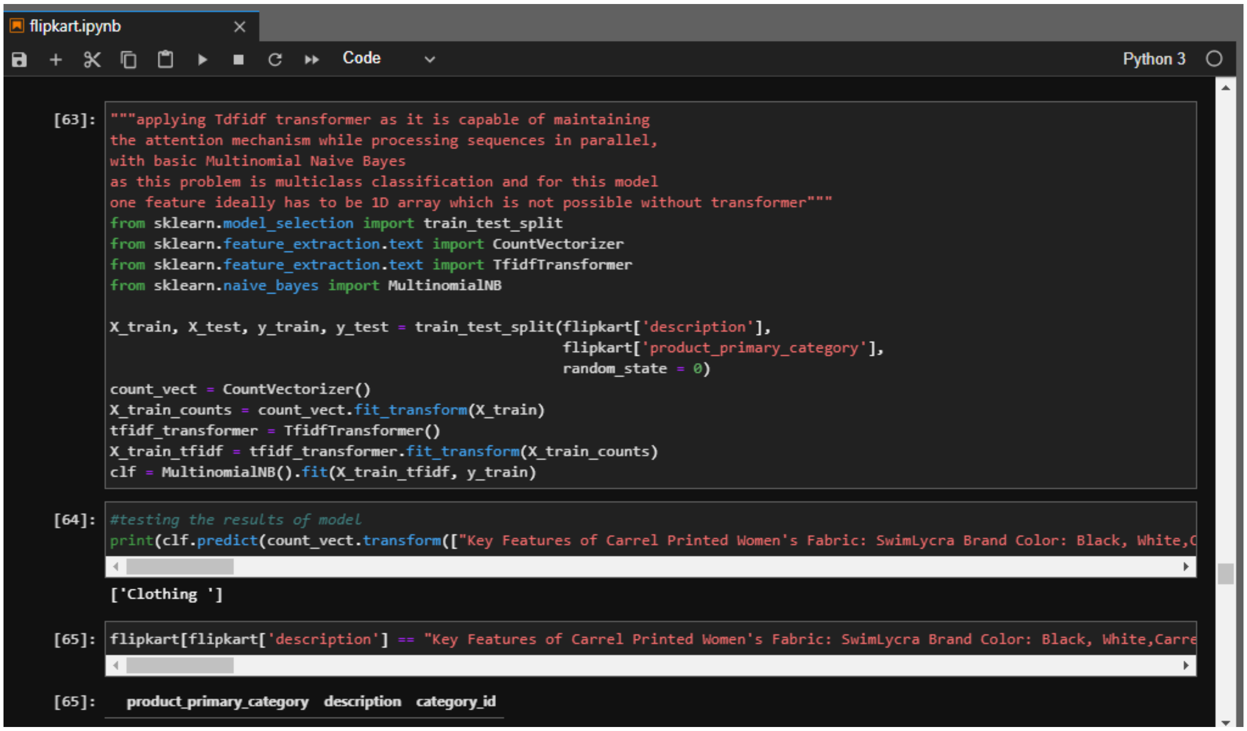
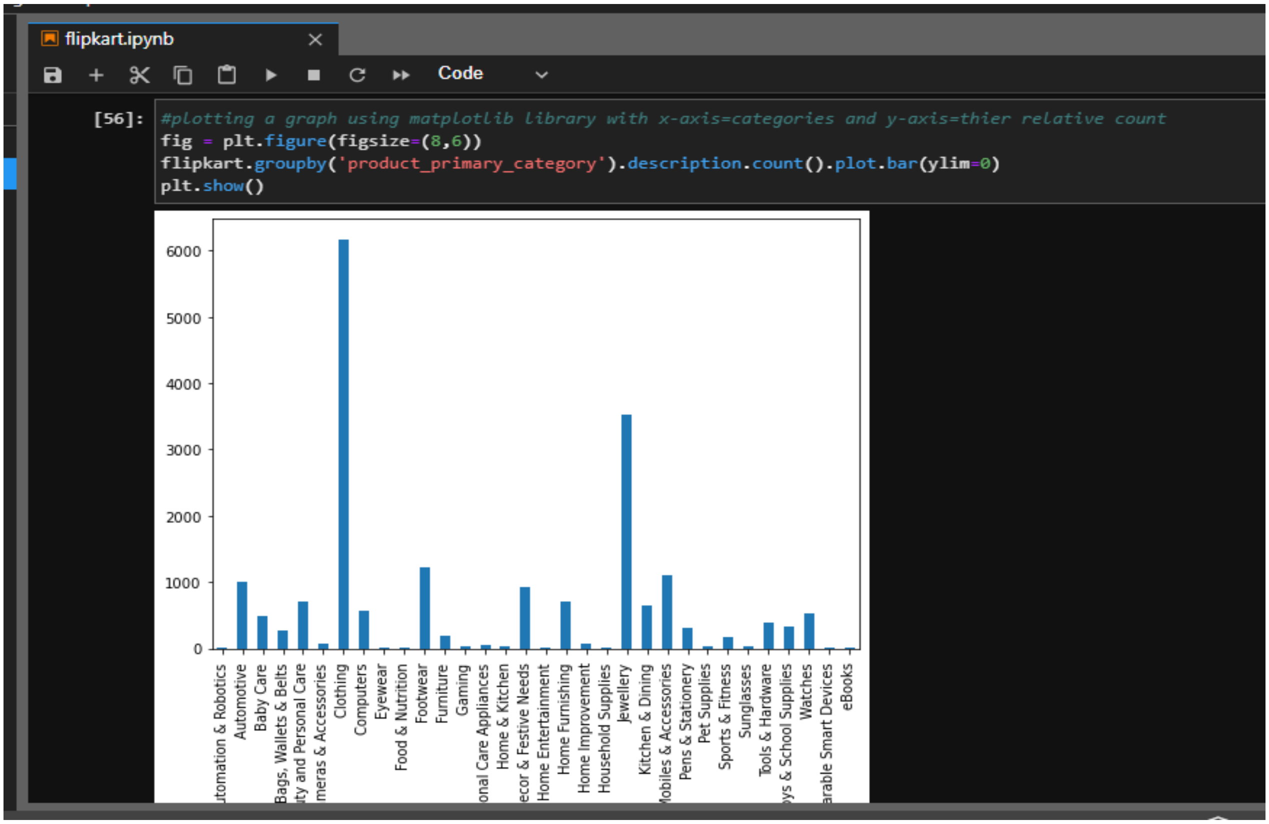
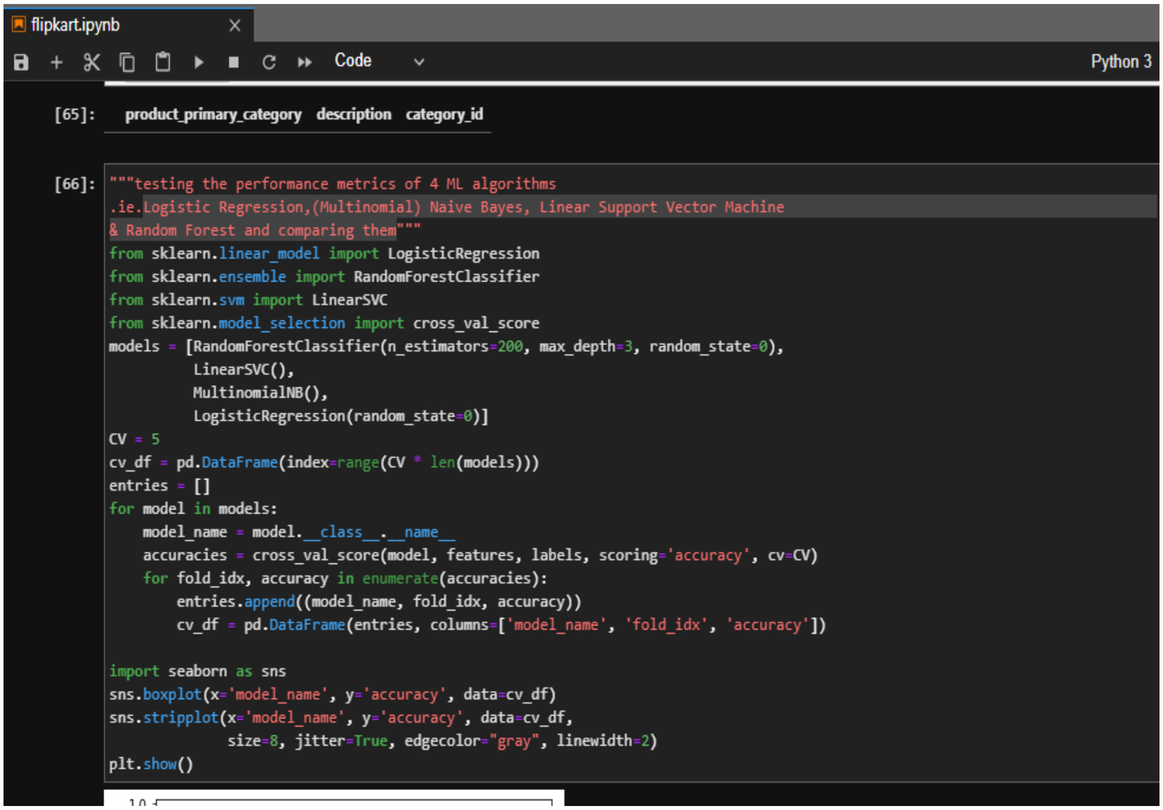


```
[63]: """applying Tdfidf transformer as it is capable of maintaining
      the attention mechanism while processing sequences in parallel,
      with basic Multinomial Naive Bayes
      as this problem is multiclass classification and for this model
      one feature ideally has to be 1D array which is not possible without transformer"""
      from sklearn.model_selection import train_test_split
      from sklearn.feature_extraction.text import CountVectorizer
      from sklearn.feature_extraction.text import TfidfTransformer
      from sklearn.naive_bayes import MultinomialNB

      X_train, X_test, y_train, y_test = train_test_split(flipkart['description'],
                                                          flipkart['product_primary_category'],
                                                          random_state = 0)
      count_vect = CountVectorizer()
      X_train_counts = count_vect.fit_transform(X_train)
      tfidf_transformer = TfidfTransformer()
      X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
      clf = MultinomialNB().fit(X_train_tfidf, y_train)

[64]: #testing the results of model
      print(clf.predict(count_vect.transform(["Key Features of Carrel Printed Women's Fabric: SwimLycra Brand Color: Black, White,C

      ['Clothing ']

[65]: flipkart[flipkart['description'] == "Key Features of Carrel Printed Women's Fabric: SwimLycra Brand Color: Black, White,Carre

[65]:   product_primary_category  description  category_id
```

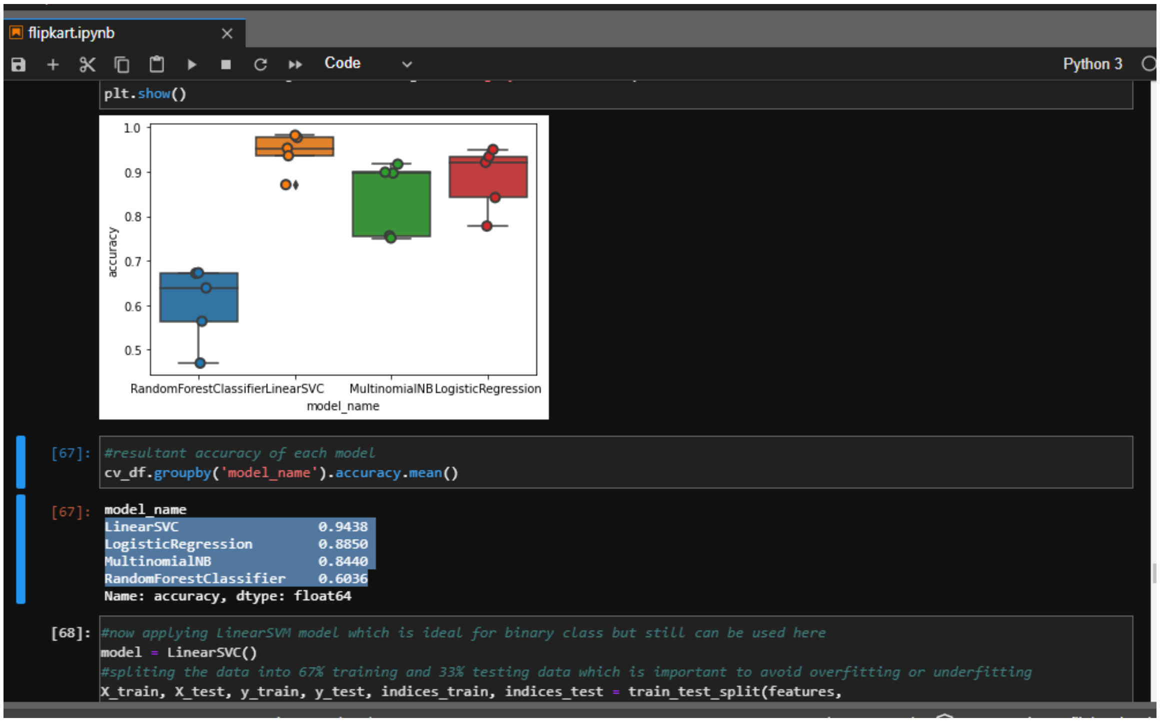


```
[56]: #plotting a graph using matplotlib library with x-axis=categories and y-axis=thier relative count
      fig = plt.figure(figsize=(8,6))
      flipkart.groupby('product_primary_category').description.count().plot.bar(ylim=0)
      plt.show()
```



```
[65]:   product_primary_category  description  category_id
```

```
[66]: """testing the performance metrics of 4 ML algorithms
.ie.Logistic Regression,(Multinomial) Naive Bayes, Linear Support Vector Machine
& Random Forest and comparing them"""
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
models = [RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
          LinearSVC(),
          MultinomialNB(),
          LogisticRegression(random_state=0)]
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))
        cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])

import seaborn as sns
sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
plt.show()
```

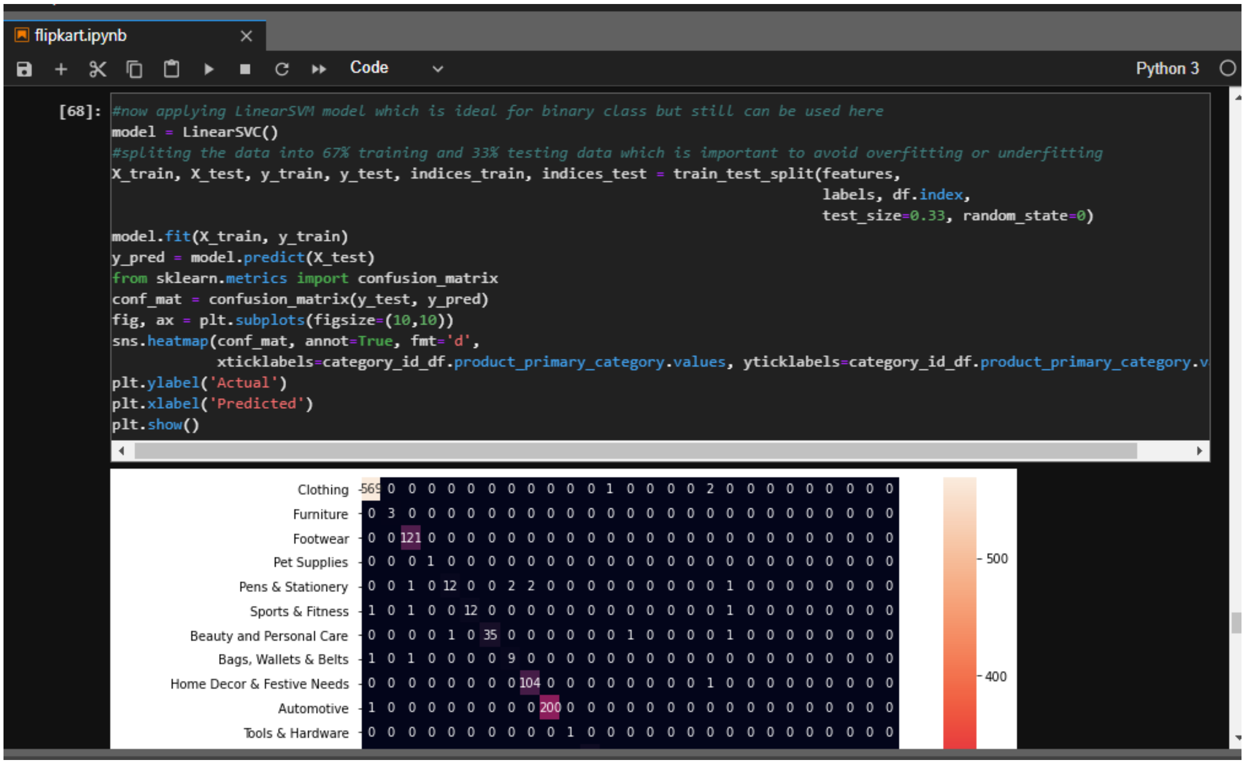


```
plt.show()
```

```
[67]: #resultant accuracy of each model
cv_df.groupby('model_name').accuracy.mean()
```

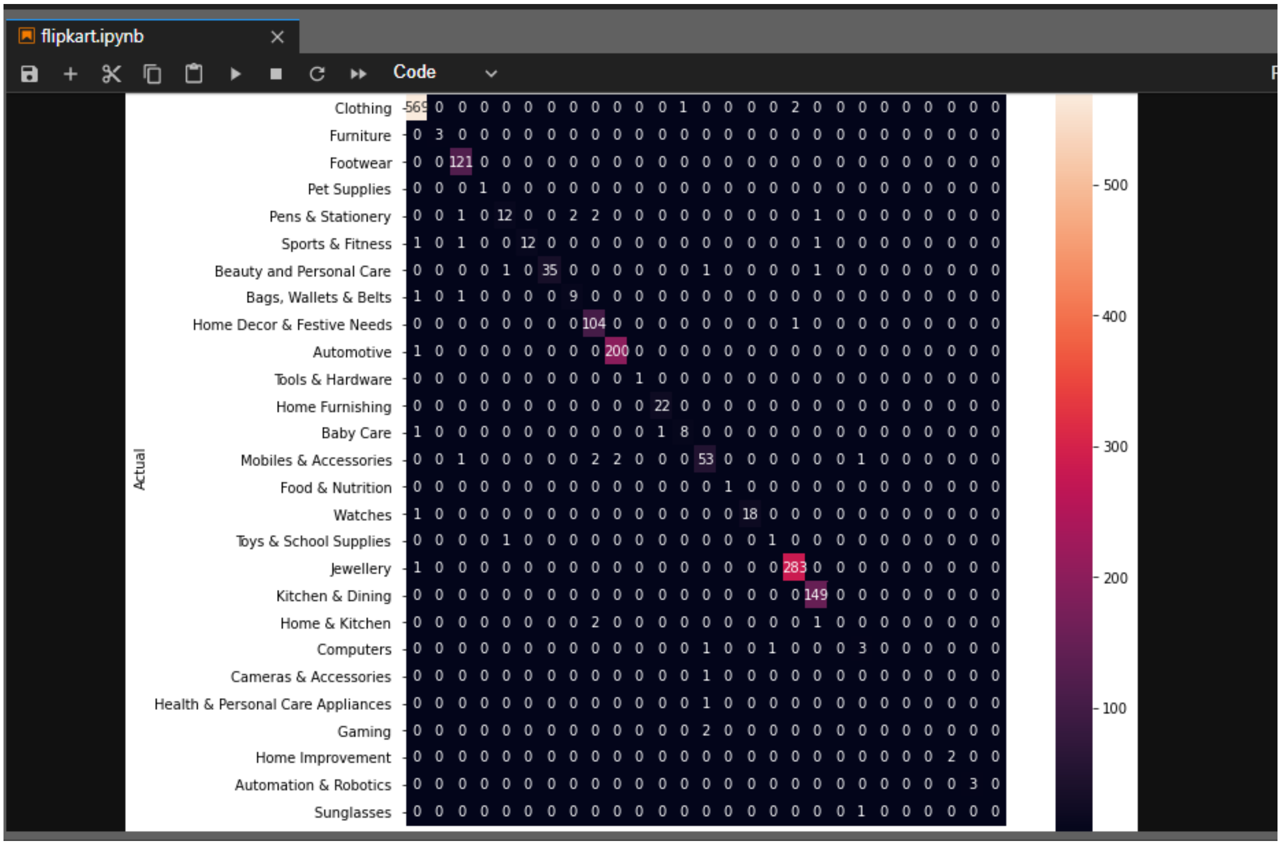
```
[67]: model_name
LinearSVC                 0.9438
LogisticRegression        0.8850
MultinomialNB             0.8440
RandomForestClassifier    0.6036
Name: accuracy, dtype: float64
```

```
[68]: #now applying LinearSVM model which is ideal for binary class but still can be used here
model = LinearSVC()
#spliting the data into 67% training and 33% testing data which is important to avoid overfitting or underfitting
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features,
```

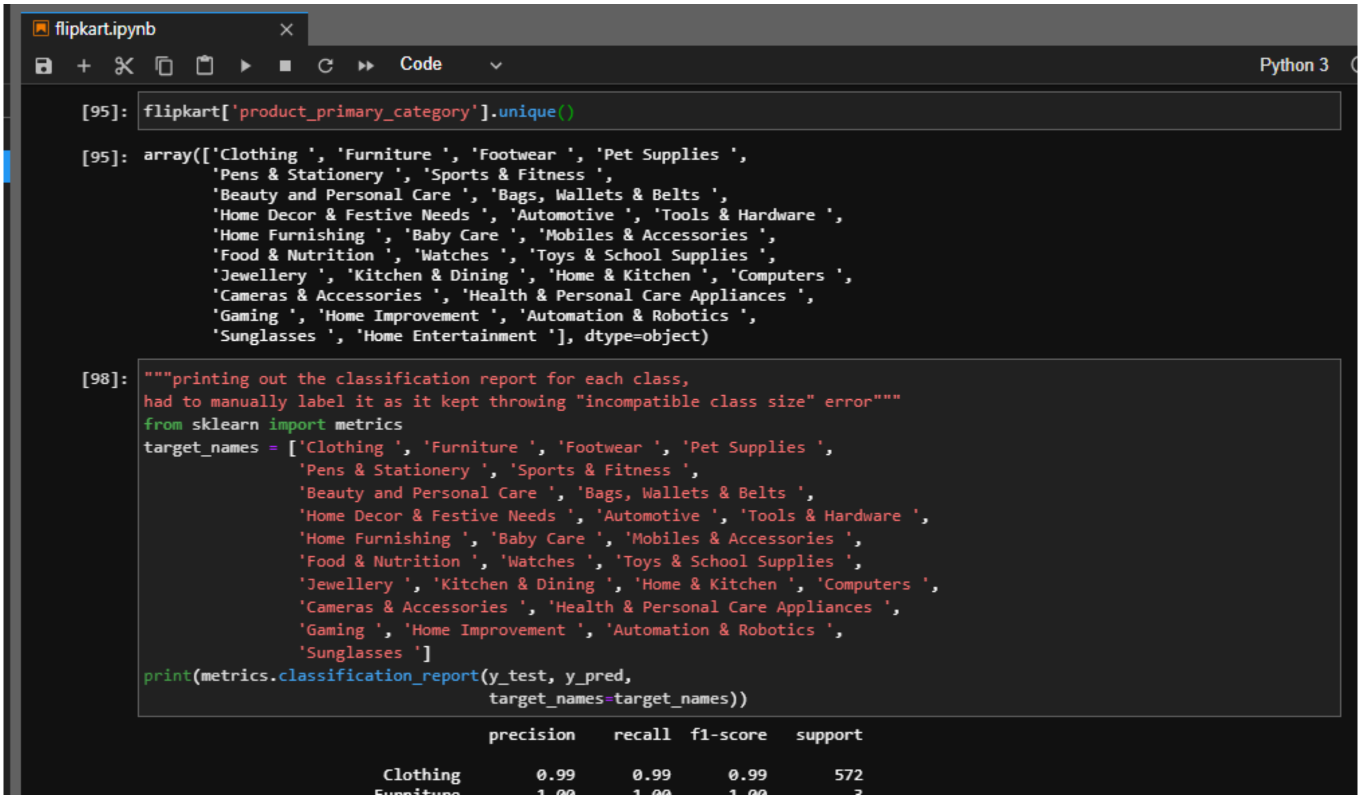


```
[68]: #now applying LinearSVM model which is ideal for binary class but still can be used here
      model = LinearSVC()
      #spliting the data into 67% training and 33% testing data which is important to avoid overfitting or underfitting
      X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features,
                                                                                       labels, df.index,
                                                                                       test_size=0.33, random_state=0)

      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      from sklearn.metrics import confusion_matrix
      conf_mat = confusion_matrix(y_test, y_pred)
      fig, ax = plt.subplots(figsize=(10,10))
      sns.heatmap(conf_mat, annot=True, fmt='d',
                  xticklabels=category_id_df.product_primary_category.values, yticklabels=category_id_df.product_primary_category.v
      plt.ylabel('Actual')
      plt.xlabel('Predicted')
      plt.show()
```

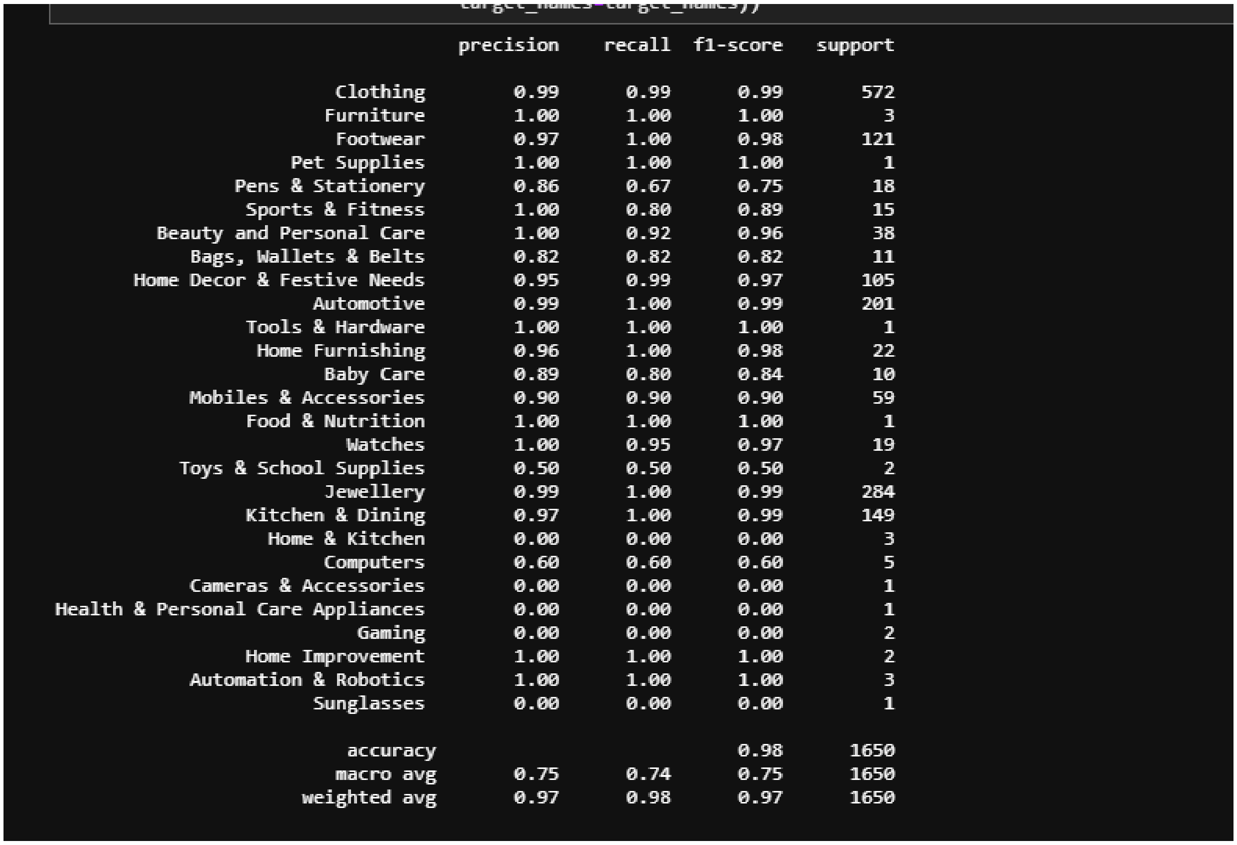
Clothing	569	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0
Furniture	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Footwear	0	0	121	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pet Supplies	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pens & Stationery	0	0	1	0	12	0	0	2	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Sports & Fitness	1	0	1	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Beauty and Personal Care	0	0	0	0	1	0	35	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
Bags, Wallets & Belts	1	0	1	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Home Decor & Festive Needs	0	0	0	0	0	0	0	0	104	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Automotive	1	0	0	0	0	0	0	0	0	200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tools & Hardware	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



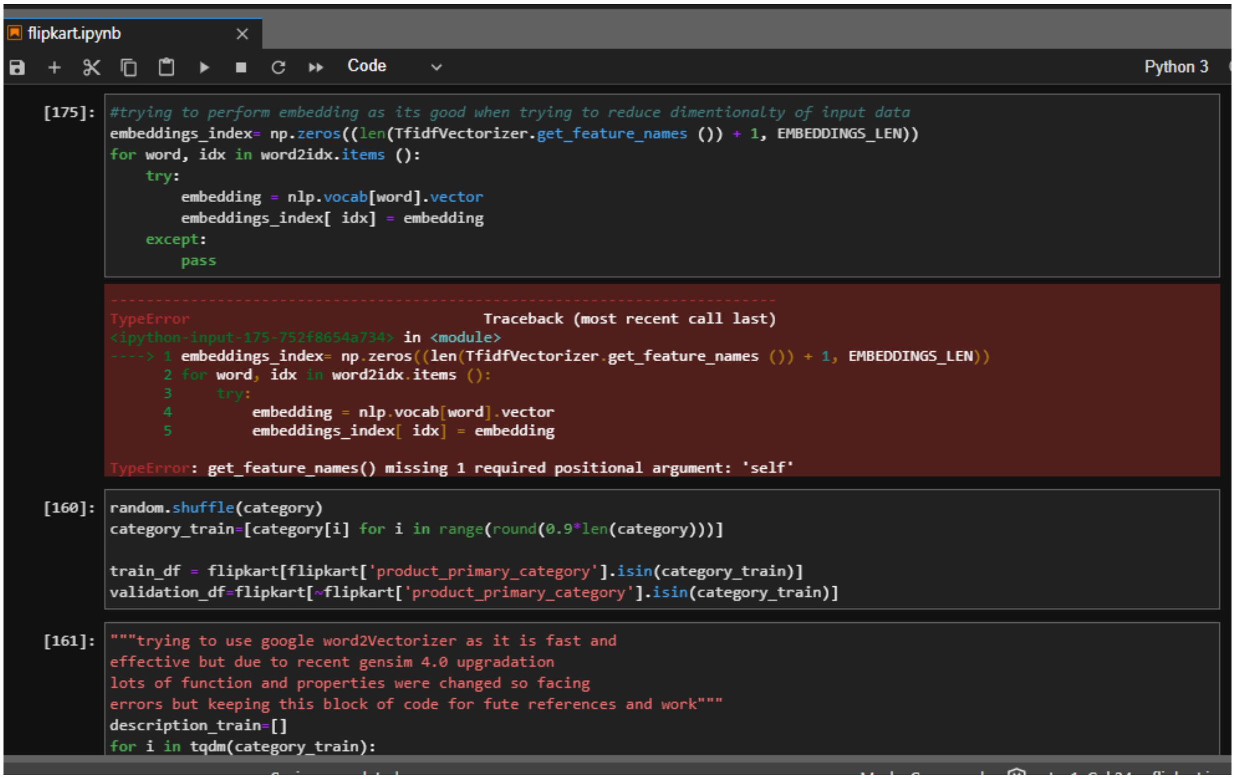
Clothing	569	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0
Furniture	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Footwear	0	0	121	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pet Supplies	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pens & Stationery	0	0	1	0	12	0	0	2	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Sports & Fitness	1	0	1	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Beauty and Personal Care	0	0	0	0	1	0	35	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
Bags, Wallets & Belts	1	0	1	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Home Decor & Festive Needs	0	0	0	0	0	0	0	0	104	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Automotive	1	0	0	0	0	0	0	0	0	200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tools & Hardware	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Home Furnishing	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Baby Care	1	0	0	0	0	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mobiles & Accessories	0	0	1	0	0	0	0	0	2	2	0	0	0	53	0	0	0	0	0	0	1	0	0	0	0	0	0
Food & Nutrition	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Watches	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0
Toys & School Supplies	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Jewellery	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	283	0	0	0	0	0	0	0	0	0
Kitchen & Dining	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	149	0	0	0	0	0	0	0	0
Home & Kitchen	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Computers	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	3	0	0	0	0	0	0
Cameras & Accessories	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Health & Personal Care Appliances	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaming	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Home Improvement	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
Automation & Robotics	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
Sunglasses	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0



```
[95]: flipkart['product_primary_category'].unique()

[95]: array(['Clothing ', 'Furniture ', 'Footwear ', 'Pet Supplies ',
              'Pens & Stationery ', 'Sports & Fitness ',
              'Beauty and Personal Care ', 'Bags, Wallets & Belts ',
              'Home Decor & Festive Needs ', 'Automotive ', 'Tools & Hardware ',
              'Home Furnishing ', 'Baby Care ', 'Mobiles & Accessories ',
              'Food & Nutrition ', 'Watches ', 'Toys & School Supplies ',
              'Jewellery ', 'Kitchen & Dining ', 'Home & Kitchen ', 'Computers ',
              'Cameras & Accessories ', 'Health & Personal Care Appliances ',
              'Gaming ', 'Home Improvement ', 'Automation & Robotics ',
              'Sunglasses ', 'Home Entertainment '], dtype=object)

[98]: """printing out the classification report for each class,
      had to manually label it as it kept throwing "incompatible class size" error"""
      from sklearn import metrics
      target_names = ['Clothing ', 'Furniture ', 'Footwear ', 'Pet Supplies ',
                      'Pens & Stationery ', 'Sports & Fitness ',
                      'Beauty and Personal Care ', 'Bags, Wallets & Belts ',
                      'Home Decor & Festive Needs ', 'Automotive ', 'Tools & Hardware ',
                      'Home Furnishing ', 'Baby Care ', 'Mobiles & Accessories ',
                      'Food & Nutrition ', 'Watches ', 'Toys & School Supplies ',
                      'Jewellery ', 'Kitchen & Dining ', 'Home & Kitchen ', 'Computers ',
                      'Cameras & Accessories ', 'Health & Personal Care Appliances ',
                      'Gaming ', 'Home Improvement ', 'Automation & Robotics ',
                      'Sunglasses ']
      print(metrics.classification_report(y_test, y_pred,
                                          target_names=target_names))

                    precision    recall  f1-score   support

          Clothing        0.99      0.99      0.99       572
```



```
                                    precision    recall  f1-score   support

                          Clothing       0.99      0.99      0.99       572
                         Furniture       1.00      1.00      1.00         3
                          Footwear       0.97      1.00      0.98       121
                      Pet Supplies       1.00      1.00      1.00         1
                 Pens & Stationery       0.86      0.67      0.75        18
                  Sports & Fitness       1.00      0.80      0.89        15
          Beauty and Personal Care       1.00      0.92      0.96        38
             Bags, Wallets & Belts       0.82      0.82      0.82        11
        Home Decor & Festive Needs       0.95      0.99      0.97       105
                        Automotive       0.99      1.00      0.99       201
                  Tools & Hardware       1.00      1.00      1.00         1
                   Home Furnishing       0.96      1.00      0.98        22
                         Baby Care       0.89      0.80      0.84        10
             Mobiles & Accessories       0.90      0.90      0.90        59
                  Food & Nutrition       1.00      1.00      1.00         1
                           Watches       1.00      0.95      0.97        19
            Toys & School Supplies       0.50      0.50      0.50         2
                         Jewellery       0.99      1.00      0.99       284
                  Kitchen & Dining       0.97      1.00      0.99       149
                    Home & Kitchen       0.00      0.00      0.00         3
                         Computers       0.60      0.60      0.60         5
             Cameras & Accessories       0.00      0.00      0.00         1
 Health & Personal Care Appliances       0.00      0.00      0.00         1
                            Gaming       0.00      0.00      0.00         2
                  Home Improvement       1.00      1.00      1.00         2
             Automation & Robotics       1.00      1.00      1.00         3
                        Sunglasses       0.00      0.00      0.00         1

                          accuracy                           0.98      1650
                         macro avg       0.75      0.74      0.75      1650
                      weighted avg       0.97      0.98      0.97      1650
```

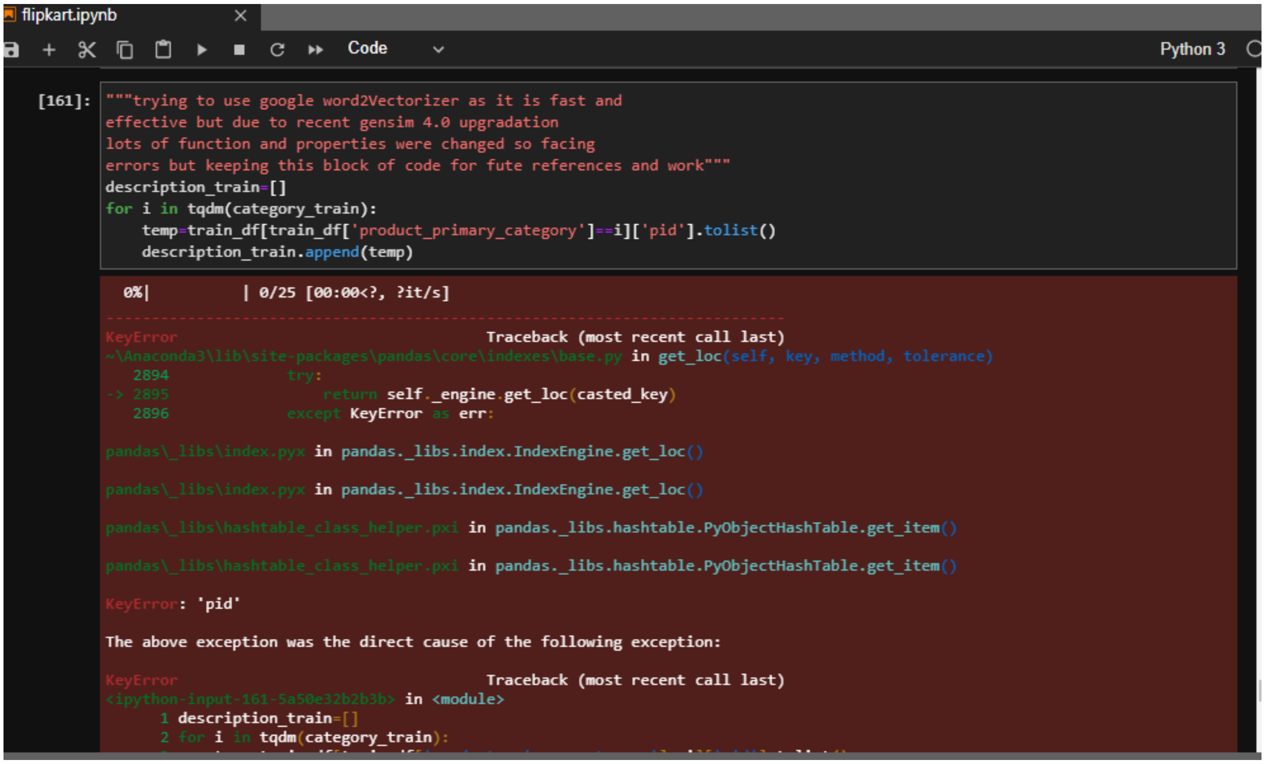


```
[175]: #trying to perform embedding as its good when trying to reduce dimentionalty of input data
       embeddings_index= np.zeros((len(TfidfVectorizer.get_feature_names ()) + 1, EMBEDDINGS_LEN))
       for word, idx in word2idx.items ():
           try:
               embedding = nlp.vocab[word].vector
               embeddings_index[ idx] = embedding
           except:
               pass
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-175-752f8654a734> in <module>
----> 1 embeddings_index= np.zeros((len(TfidfVectorizer.get_feature_names ()) + 1, EMBEDDINGS_LEN))
      2 for word, idx in word2idx.items ():
      3     try:
      4         embedding = nlp.vocab[word].vector
      5         embeddings_index[ idx] = embedding

TypeError: get_feature_names() missing 1 required positional argument: 'self'
```

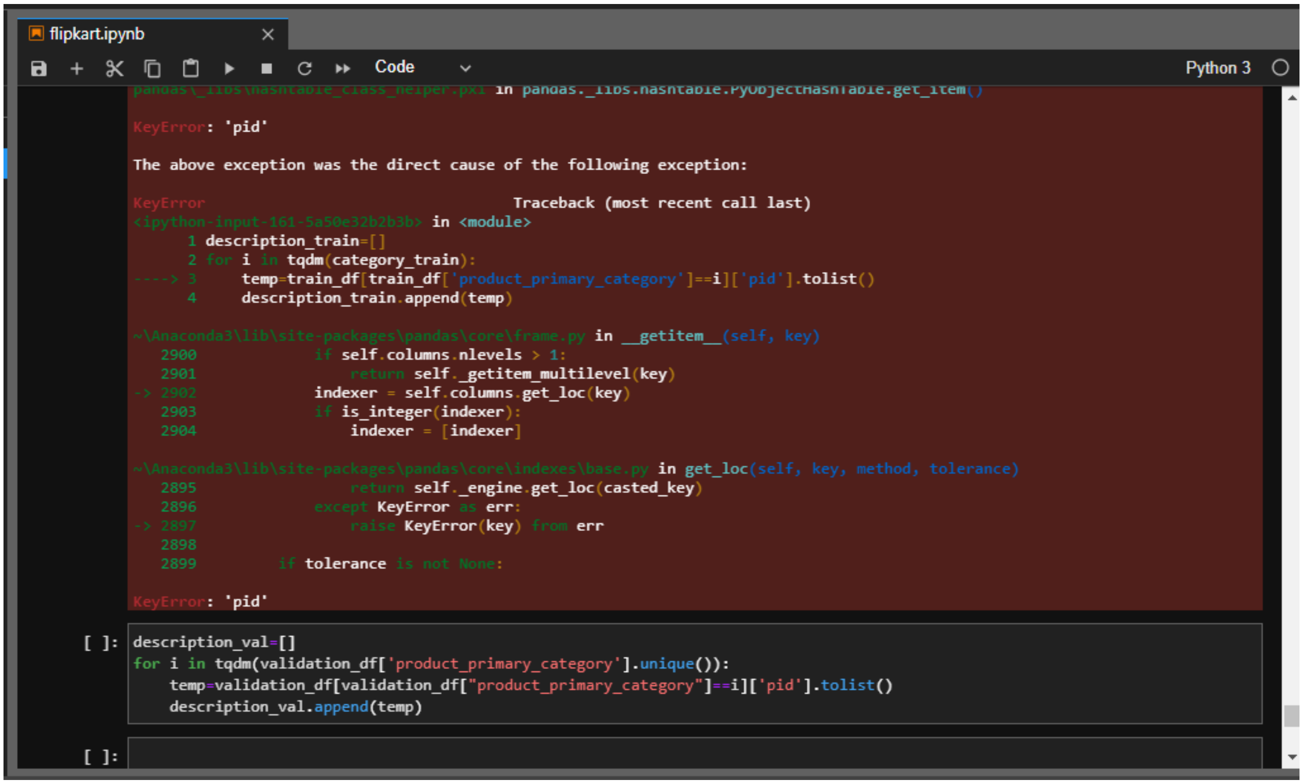
```
[160]: random.shuffle(category)
       category_train=[category[i] for i in range(round(0.9*len(category)))]

       train_df = flipkart[flipkart['product_primary_category'].isin(category_train)]
       validation_df=flipkart[~flipkart['product_primary_category'].isin(category_train)]
```

```
[161]: """trying to use google word2Vectorizer as it is fast and
       effective but due to recent gensim 4.0 upgradation
       lots of function and properties were changed so facing
       errors but keeping this block of code for fute references and work"""
       description_train=[]
       for i in tqdm(category_train):
```



```
[161]: """trying to use google word2Vectorizer as it is fast and
       effective but due to recent gensim 4.0 upgradation
       lots of function and properties were changed so facing
       errors but keeping this block of code for fute references and work"""
       description_train=[]
       for i in tqdm(category_train):
           temp=train_df[train_df['product_primary_category']==i]['pid'].tolist()
           description_train.append(temp)
```

```
  0%|          | 0/25 [00:00<?, ?it/s]
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2894             try:
-> 2895                 return self._engine.get_loc(casted_key)
   2896             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'pid'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-161-5a50e32b2b3b> in <module>
      1 description_train=[]
      2 for i in tqdm(category_train):
```



```
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'pid'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-161-5a50e32b2b3b> in <module>
      1 description_train=[]
      2 for i in tqdm(category_train):
----> 3     temp=train_df[train_df['product_primary_category']==i]['pid'].tolist()
      4     description_train.append(temp)

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2900             if self.columns.nlevels > 1:
   2901                 return self._getitem_multilevel(key)
-> 2902             indexer = self.columns.get_loc(key)
   2903             if is_integer(indexer):
   2904                 indexer = [indexer]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2895                 return self._engine.get_loc(casted_key)
   2896             except KeyError as err:
-> 2897                 raise KeyError(key) from err
   2898 
   2899         if tolerance is not None:

KeyError: 'pid'
```

```
[ ]: description_val=[]
     for i in tqdm(validation_df['product_primary_category'].unique()):
         temp=validation_df[validation_df["product_primary_category"]==i]['pid'].tolist()
         description_val.append(temp)

[ ]:
```