Analysis Report

Lab Assignment 2: Classification

Garnett Grant

COMP 247 – W2024

Mohammad Saiful Islam

January 20th, 2024

8. “Use imshow method to plot the values of the three variables you defined in the above point. Note the values in your Analysis report (written response).”

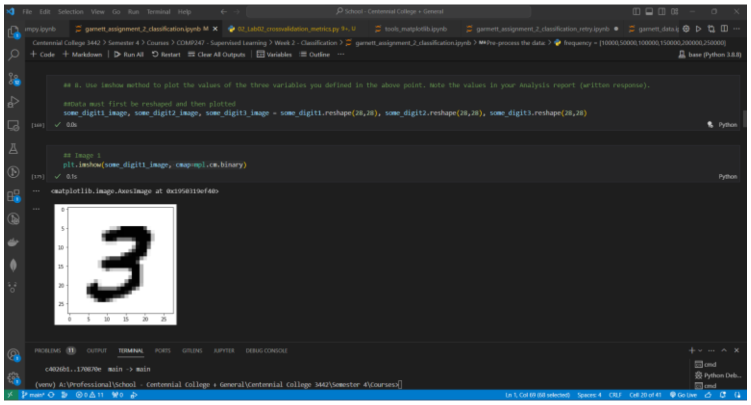
When attempting to use the “imshow” method to plot the raw values of the three variables defined from question 7, the system throws an error stating:

```
TypeError: Invalid shape (784,) for image data
```

Based on this error, the raw data has to be reshaped first and then plotted.

After reshaping the raw data into a 28x28 image, a legible image is displayed.

This can be further fit for analysis by including the cmap attribute and changing the color map display type to binary (i.e., black and white or grayscale)

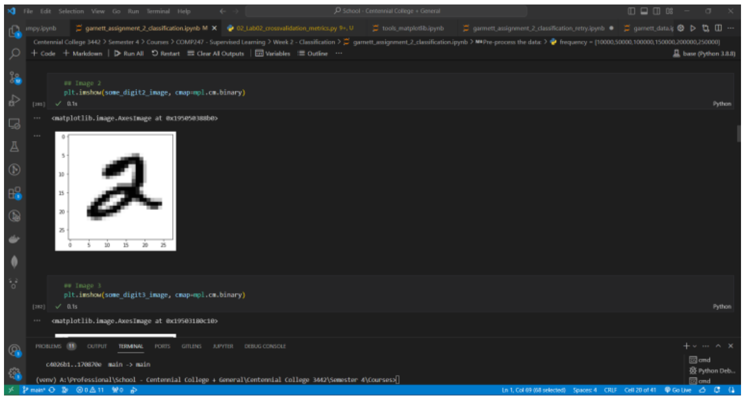


```
## 8. Use imshow method to plot the values of the three variables you defined in the above point. Note the values in your Analysis report (written response).

##Data must first be reshaped and then plotted
some_digit1_image, some_digit2_image, some_digit3_image = some_digit1.reshape(28,28), some_digit2.reshape(28,28), some_digit3.reshape(28,28)
```

```
## Image 1
plt.imshow(some_digit1_image, cmap=mpl.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x1950319ef40>
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```



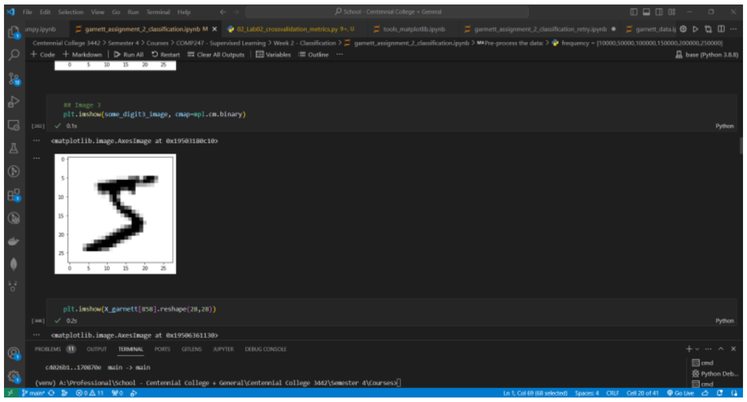
```
## Image 2
plt.imshow(some_digit2_image, cmap=mpl.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x195050388b0>
```

```
## Image 3
plt.imshow(some_digit3_image, cmap=mpl.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x19503180c10>
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```



```
## Image 3
plt.imshow(some_digit3_image, cmap=mpl.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x19503180c10>
```

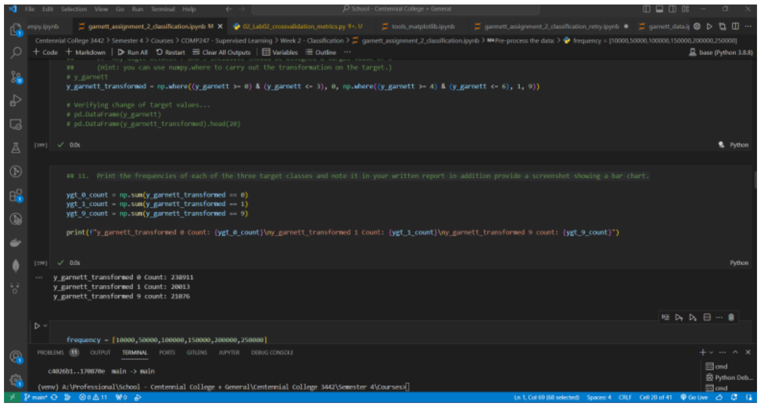
```
plt.imshow(X_garnett[858].reshape(28,28))
```

```
<matplotlib.image.AxesImage at 0x19506361130>
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

11. Print the frequencies of each of the three target classes and note it in your written report in addition provide a screenshot showing a bar chart.

The frequencies of each of the three target classes are vastly different when comparing the count of the 0 classes vs the other two classes, 1 & 9 respectively.



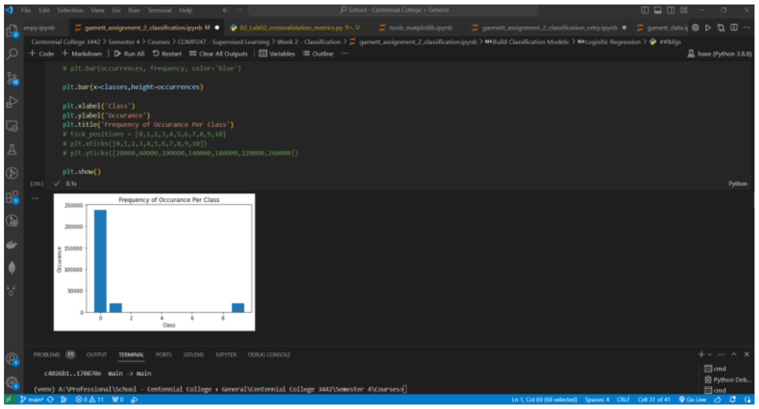
```
##     (Hint: you can use numpy.where to carry out the transformation on the target.)
# y_garnett
y_garnett_transformed = np.where((y_garnett >= 0) & (y_garnett <= 3), 0, np.where((y_garnett >= 4) & (y_garnett <= 6), 1, 9))

# Verifying change of target values...
# pd.DataFrame(y_garnett)
# pd.DataFrame(y_garnett_transformed).head(20)
```

```
## 11.  Print the frequencies of each of the three target classes and note it in your written report in addition provide a screenshot showing a bar chart.

ygt_0_count = np.sum(y_garnett_transformed == 0)
ygt_1_count = np.sum(y_garnett_transformed == 1)
ygt_9_count = np.sum(y_garnett_transformed == 9)

print(f"y_garnett_transformed 0 Count: {ygt_0_count}\ny_garnett_transformed 1 Count: {ygt_1_count}\ny_garnett_transformed 9 count: {ygt_9_count}")
```

```
y_garnett_transformed 0 Count: 238911
y_garnett_transformed 1 Count: 20013
y_garnett_transformed 9 count: 21076
```

```
frequency = [10000,50000,100000,150000,200000,250000]
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

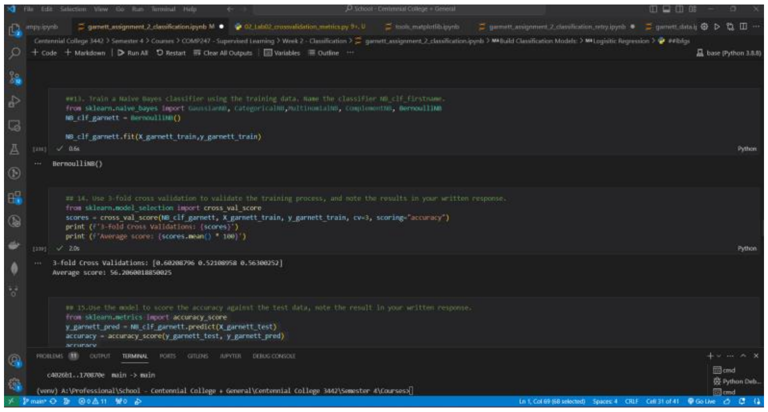


```
# plt.bar(occurrences, frequency, color='blue')

plt.bar(x=classes,height=occurrences)

plt.xlabel('Class')
plt.ylabel('Occurance')
plt.title('Frequency of Occurance Per Class')
# tick_positions = [0,1,2,3,4,5,6,7,8,9,10]
# plt.xticks([0,1,2,3,4,5,6,7,8,9,10])
# plt.yticks([20000,60000,100000,140000,180000,220000,260000])

plt.show()
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

14. Use 3-fold cross validation to validate the training process, and note the results in your written response.

Upon using 3-fold cross validation to validate the training process, the results are conclusively inefficient. The average score for the model is 56%. The score also observingly fluctuates through lower percentages after each cross-validation execution.



```
##13. Train a Naive Bayes classifier using the training data. Name the classifier NB_clf_firstname.
from sklearn.naive_bayes import GaussianNB, CategoricalNB,MultinomialNB, ComplementNB, BernoulliNB
NB_clf_garnett = BernoulliNB()

NB_clf_garnett.fit(X_garnett_train,y_garnett_train)
```

```
BernoulliNB()
```

```
## 14. Use 3-fold cross validation to validate the training process, and note the results in your written response.
from sklearn.model_selection import cross_val_score
scores = cross_val_score(NB_clf_garnett, X_garnett_train, y_garnett_train, cv=3, scoring="accuracy")
print (f'3-fold Cross Validations: {scores}')
print (f'Average score: {scores.mean() * 100}')
```

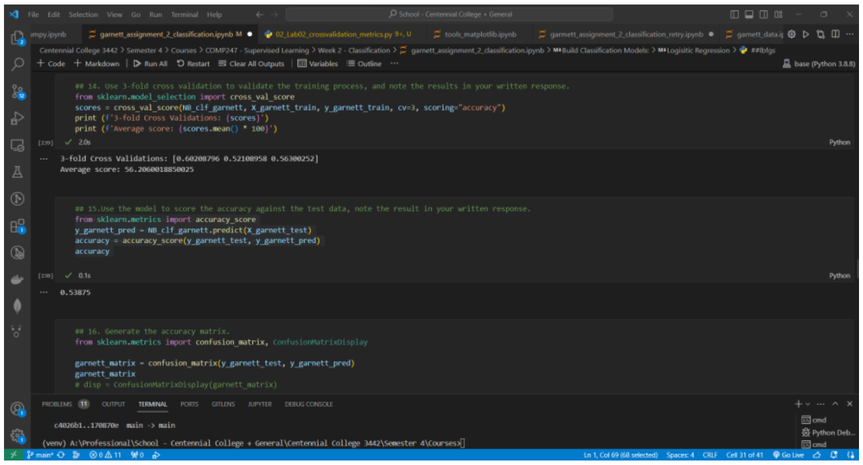
```
3-fold Cross Validations: [0.60208796 0.52108958 0.56300252]
Average score: 56.2060018850025
```

```
## 15.Use the model to score the accuracy against the test data, note the result in your written response.
from sklearn.metrics import accuracy_score
y_garnett_pred = NB_clf_garnett.predict(X_garnett_test)
accuracy = accuracy_score(y_garnett_test, y_garnett_pred)
accuracy
```

```
c4026b1..170870e  main -> main
(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

15. Use the model to score the accuracy against the test data, note the result in your written response.

The accuracy against the test data is calculated at 53% (54% rounded). This is an unfavourable outcome.



```
## 14. Use 3-fold cross validation to validate the training process, and note the results in your written response.
from sklearn.model_selection import cross_val_score
scores = cross_val_score(NB_clf_garnett, X_garnett_train, y_garnett_train, cv=3, scoring="accuracy")
print (f'3-fold Cross Validations: {scores}')
print (f'Average score: {scores.mean() * 100}')
```

```
3-fold Cross Validations: [0.60208796 0.52108958 0.56300252]
Average score: 56.2060018850025
```

```
## 15.Use the model to score the accuracy against the test data, note the result in your written response.
from sklearn.metrics import accuracy_score
y_garnett_pred = NB_clf_garnett.predict(X_garnett_test)
accuracy = accuracy_score(y_garnett_test, y_garnett_pred)
accuracy
```

```
0.53875
```

```
## 16. Generate the accuracy matrix.
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

garnett_matrix = confusion_matrix(y_garnett_test, y_garnett_pred)
garnett_matrix
# disp = ConfusionMatrixDisplay(garnett_matrix)
```

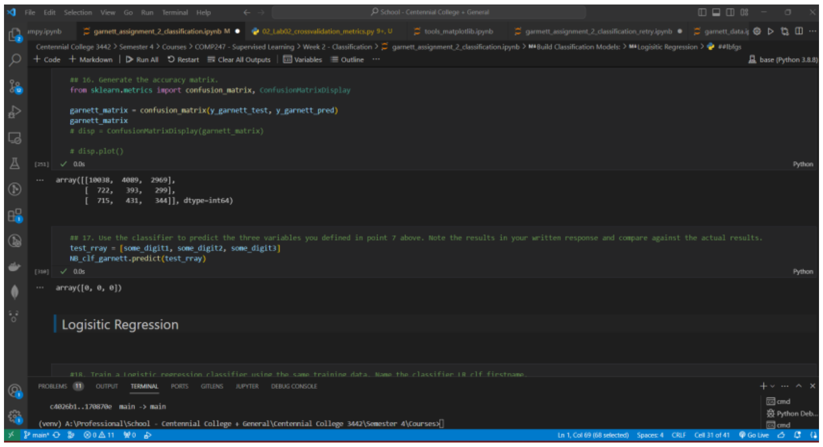
```
c4026b1..170870e  main -> main
(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

17. Use the classifier to predict the three variables you defined in point 7 above. Note the results in your written response and compare against the actual results.

Upon using the classifier to predict the three variables defined in point 7 (some_digit1, some_digit2, some_digit3), the classifier predicted these classes as all 0s. 2/3 were correct.

some_digit1 is 3, some_digit2 is 2, and some_digit3 is 5.

Considering the target variables have been transformed into 3 classes (0 made up of digits in between 0 and 3 inclusive, 1 made up of digits in between 4 and 6 inclusive, & 9 made up of digits in between 7 and 9 inclusive), only 2/3 of the predictions are correct.


```
## 16. Generate the accuracy matrix.
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

garnett_matrix = confusion_matrix(y_garnett_test, y_garnett_pred)
garnett_matrix
# disp = ConfusionMatrixDisplay(garnett_matrix)

# disp.plot()
```

```
array([[10038,  4089,  2969],
       [  722,   393,   299],
       [  715,   431,   344]], dtype=int64)
```

```
## 17. Use the classifier to predict the three variables you defined in point 7 above. Note the results in your written response and compare against the actual results.
test_rray = [some_digit1, some_digit2, some_digit3]
NB_clf_garnett.predict(test_rray)
```

```
array([0, 0, 0])
```

Logisitic Regression

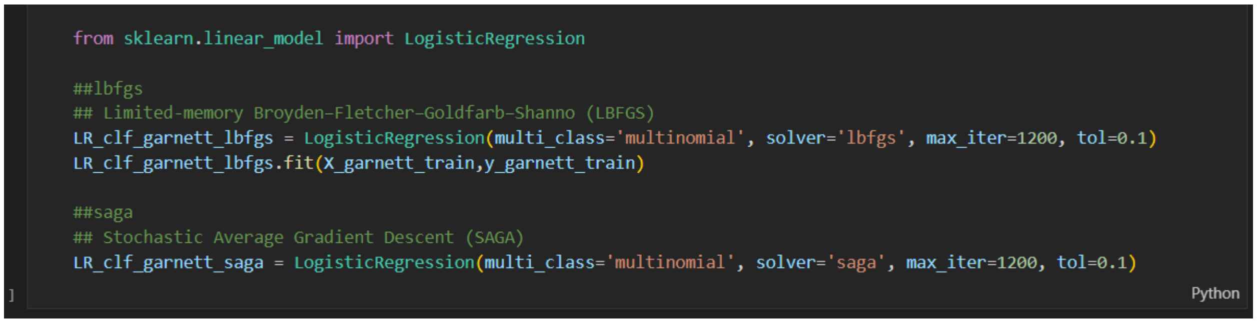
```
#18. Train a Logistic regression classifier using the same training data. Name the classifier LR_clf_firstname.
```

```
 c4026b1..170870e  main -> main
(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```
File
Edit
Selection
View
Go
Run
Terminal
Help
School - Centennial College + General
garnett_assignment_2_classification.ipynb
02_Lab02_crossvalidation_metrics.py
tools_matplotlib.ipynb
garnett_assignment_2_classification_retry.ipynb
Centennial College 3442 > Semester 4 > Courses > COMP247 - Supervised Learning > Week 2 - Classification > garnett_assignment_2_classification.ipynb > Build Classification Models: > Logisitic Regression > ##lbfgs
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.8.8)
Python
PROBLEMS
OUTPUT
TERMINAL
PORTS
GITLENS
JUPYTER
DEBUG CONSOLE
cmd
Python Deb...
Ln 1, Col 69 (68 selected)
Spaces: 4
CRLF
Cell 31 of 41
Go Live

18. Train a Logistic regression classifier using the same training data. Name the classifier LR_clf_firstname.

(Note this is a multi-class problem make sure to check all the parameters and set multi_class='multinomial').

Try training the classifier using two solvers first "lbfgs" then "Saga". Set max_iter to 1200 and tolerance to 0.1 in both cases.



```python
from sklearn.linear_model import LogisticRegression

##lbfgs
## Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS)
LR_clf_garnett_lbfgs = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1200, tol=0.1)
LR_clf_garnett_lbfgs.fit(X_garnett_train,y_garnett_train)

##saga
## Stochastic Average Gradient Descent (SAGA)
LR_clf_garnett_saga = LogisticRegression(multi_class='multinomial', solver='saga', max_iter=1200, tol=0.1)
```

Make sure you note the results in both cases in your written response, and note the main differences in your #written response.

Carryout a quick research on the difference between the "lbfgs" and "Saga" solvers and see #how this applies to the results, note that size and dimensions of the dataset. Don't worry if one doesn't converge your research should explain why. Note the results of your research in your analysis report.

<u>lbfgs</u>

lbfgs stands for (Limited-memory Broyden-Fletcher-Goldfarb-Shanno). Lbfgs is a solver that has 3 notable features. Firstly, it's another form of a known solver coined as Newton's Method (newton-cg). Secondly, it utilizes limited memory i.e., it retains some vectors that demonstrate estimations non-explicitly. Thirdly, it can perform at a more desired rate on smaller datasets which in turn will save more memory. Some notable flaws of lbfgs require more attention to detail as this solver can eventually converge to nothing.

When using lbfgs solver for my classification model, I observed that the time it took to fit the training data was quite long. It took a staggering 3 mins (+ -). It took relatively the same amount of time just to cross validate the score. Although the time of execution was longer, it has been proven to be quite successful with an accuracy of 85%.

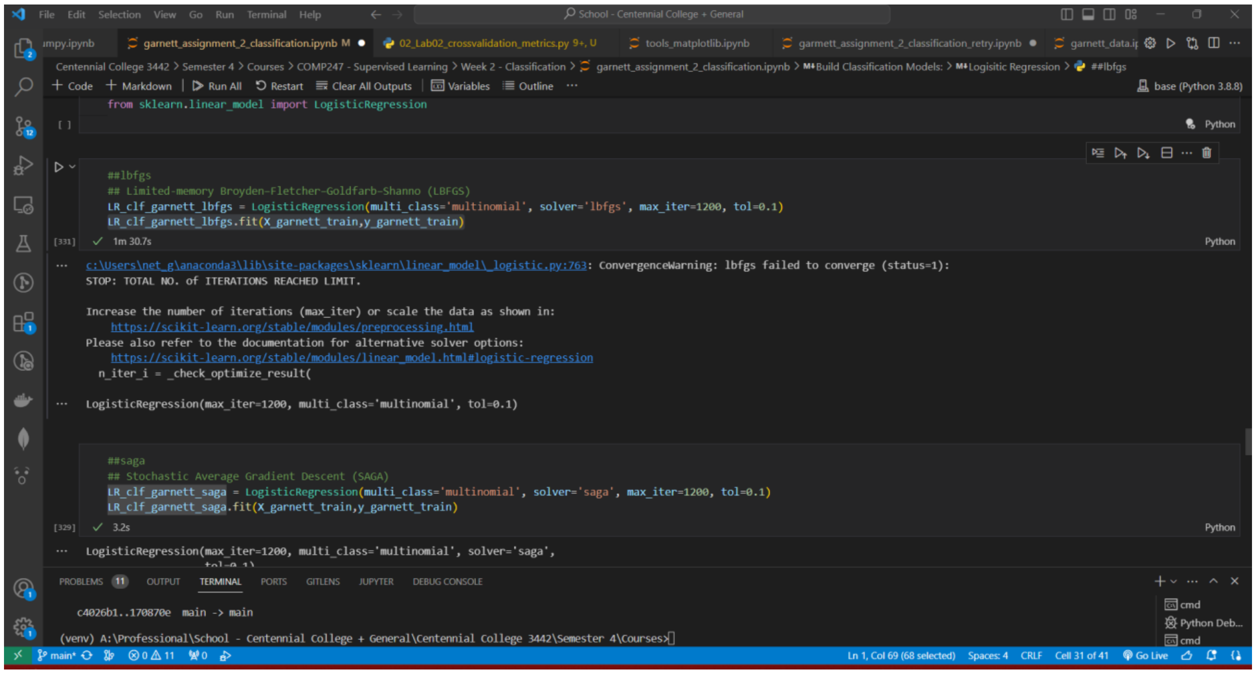
<u>Saga</u>

Saga stands for (Stochastic Average Gradient Descent). Saga is a solver that supports all penalties that the other solvers do. While it does have a memory cost of O(n), it is still computationally faster than other solvers for large datasets. It is regularly denoted as one of the better choices for a solver.

When utilizing saga, I observed that the time it took to fit the training data was amazingly swift. This solver completed the task in approximately 3.2s! The time it took to calculate the cross-validation score was also amazing, only taking 6.8s! What's even more impressive is it's accuracy of 85%. While the accuracy is similar when utilizing the lbfgs solver, the time of execution offers a more desired overall performance.

<u>Comparison</u>

In conclusion, while lbfgs and saga both have considerably high accuracy, both have their own flaws that need to be considered. Nevertheless, when it comes to large datasets like the mnist_784 dataset, saga is the better solver for computational time and accuracy.


File Edit Selection View Go Run Terminal Help
School - Centennial College + General
garnett_assignment_2_classification.ipynb M
02_Lab02_crossvalidation_metrics.py 9+, U
tools_matplotlib.ipynb
garnett_assignment_2_classification_retry.ipynb
Centennial College 3442 > Semester 4 > Courses > COMP247 - Supervised Learning > Week 2 - Classification > garnett_assignment_2_classification.ipynb > Build Classification Models: > Logisitic Regression > ##lbfgs
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.8.8)

```
from sklearn.linear_model import LogisticRegression
```

```
##lbfgs
## Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS)
LR_clf_garnett_lbfgs = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1200, tol=0.1)
LR_clf_garnett_lbfgs.fit(X_garnett_train,y_garnett_train)
```

```
[331] ✓ 1m 30.7s

c:\Users\net_g\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

LogisticRegression(max_iter=1200, multi_class='multinomial', tol=0.1)
```

```
##saga
## Stochastic Average Gradient Descent (SAGA)
LR_clf_garnett_saga = LogisticRegression(multi_class='multinomial', solver='saga', max_iter=1200, tol=0.1)
LR_clf_garnett_saga.fit(X_garnett_train,y_garnett_train)
```

```
[329] ✓ 3.2s

LogisticRegression(max_iter=1200, multi_class='multinomial', solver='saga',
```

PROBLEMS 11 OUTPUT TERMINAL PORTS GITLENS JUPYTER DEBUG CONSOLE

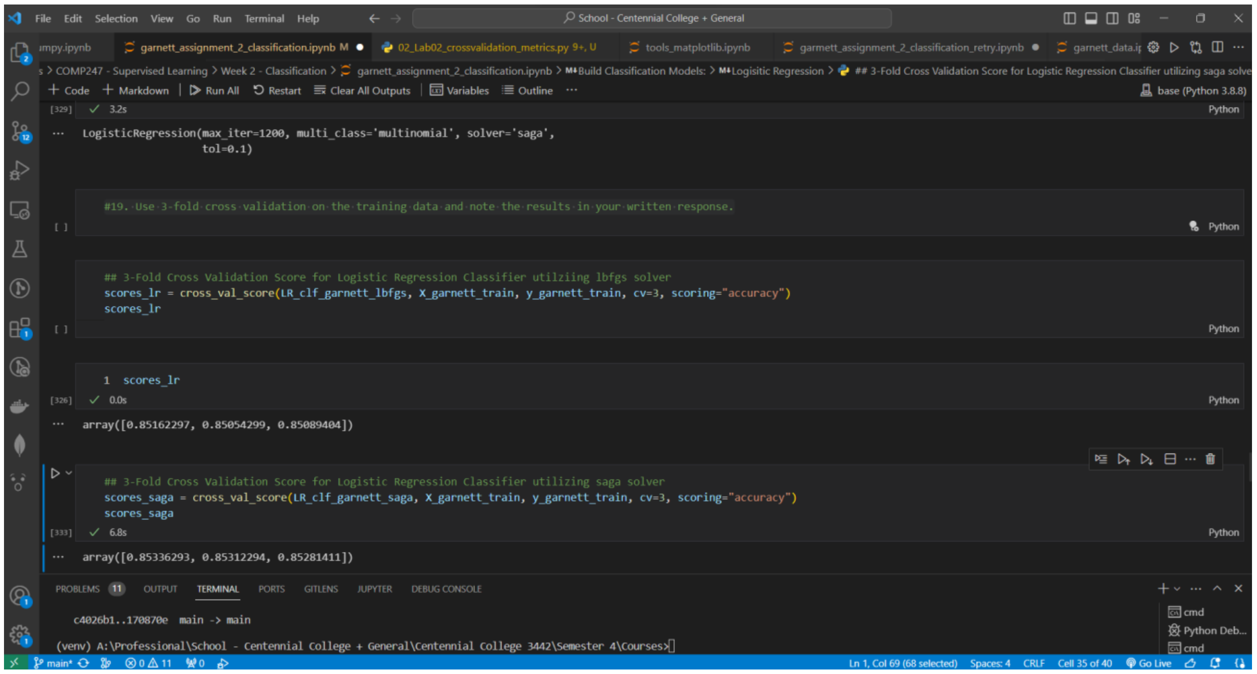
```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

main* 0 11 0
Ln 1, Col 69 (68 selected) Spaces: 4 CRLF Cell 31 of 41 Go Live

19. Use 3-fold cross validation on the training data and note the results in your written response.

As mentioned above, the 3-fold cross validation on the Logistic Regression Classifier using the lbfgs solver is very accurate at an acceptable 85%. When contrasting this performance to the initial Naïve Bayes Classifier used in Question 13, it is remarkable. It did however, take 2m and 50s just to execute.

The 3-fold cross validation on the Logistic Regression Classifier using the saga solver is also desirably accurate at a decent 85.3%.


File Edit Selection View Go Run Terminal Help
School - Centennial College + General
garnett_assignment_2_classification.ipynb M
02_Lab02_crossvalidation_metrics.py 9+, U
tools_matplotlib.ipynb
garnett_assignment_2_classification_retry.ipynb
COMP247 - Supervised Learning > Week 2 - Classification > garnett_assignment_2_classification.ipynb > Build Classification Models: > Logisitic Regression > ## 3-Fold Cross Validation Score for Logistic Regression Classifier utilizing saga solve
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.8.8)
Python

```
[329] ✓ 3.2s
LogisticRegression(max_iter=1200, multi_class='multinomial', solver='saga',
                   tol=0.1)
```

```
#19. Use 3-fold cross validation on the training data and note the results in your written response.
```

```
## 3-Fold Cross Validation Score for Logistic Regression Classifier utilziing lbfgs solver
scores_lr = cross_val_score(LR_clf_garnett_lbfgs, X_garnett_train, y_garnett_train, cv=3, scoring="accuracy")
scores_lr
```

```
scores_lr
[326] ✓ 0.0s
array([0.85162297, 0.85054299, 0.85089404])
```

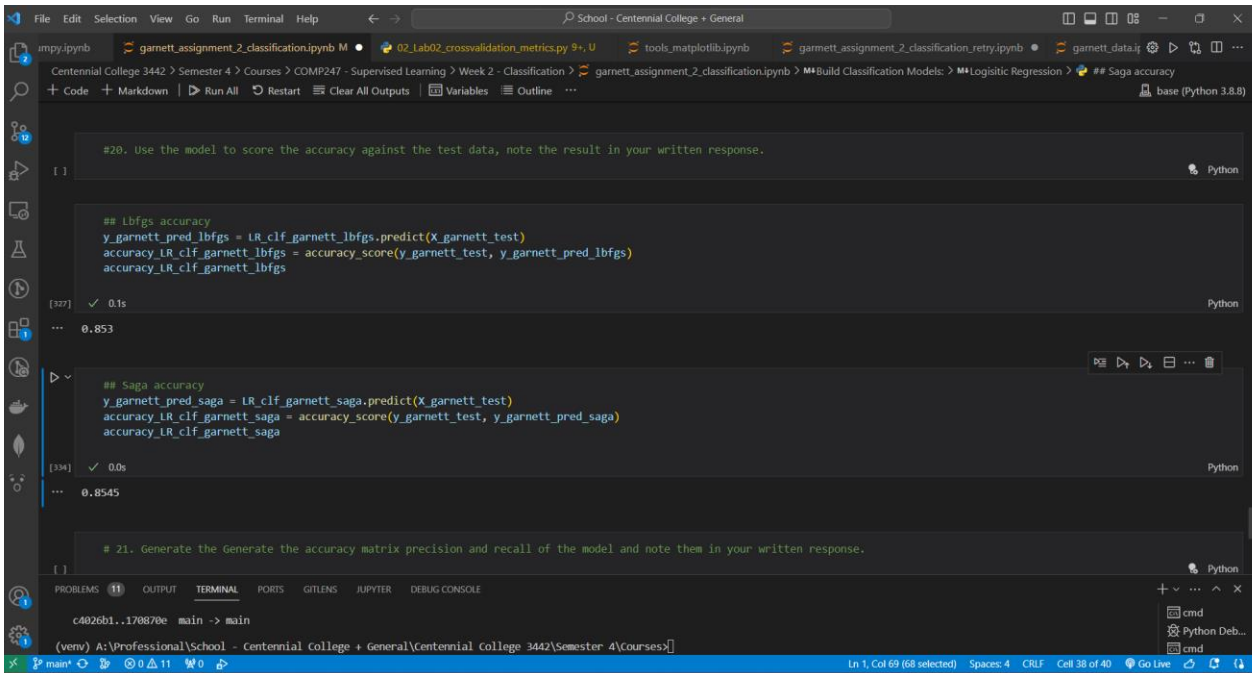
```
## 3-Fold Cross Validation Score for Logistic Regression Classifier utilizing saga solver
scores_saga = cross_val_score(LR_clf_garnett_saga, X_garnett_train, y_garnett_train, cv=3, scoring="accuracy")
scores_saga
[333] ✓ 6.8s
array([0.85336293, 0.85312294, 0.85281411])
```

PROBLEMS 11 OUTPUT TERMINAL PORTS GITLENS JUPYTER DEBUG CONSOLE

```
c4026b1..170870e  main -> main
(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

main* ⊗0 ⚠11 Ln 1, Col 69 (68 selected) Spaces: 4 CRLF Cell 35 of 40 Go Live

20. Use the model to score the accuracy against the test data, note the result in your written response.



```
#20. Use the model to score the accuracy against the test data, note the result in your written response.
```

```
## Lbfgs accuracy
y_garnett_pred_lbfgs = LR_clf_garnett_lbfgs.predict(X_garnett_test)
accuracy_LR_clf_garnett_lbfgs = accuracy_score(y_garnett_test, y_garnett_pred_lbfgs)
accuracy_LR_clf_garnett_lbfgs
```

```
0.853
```

```
## Saga accuracy
y_garnett_pred_saga = LR_clf_garnett_saga.predict(X_garnett_test)
accuracy_LR_clf_garnett_saga = accuracy_score(y_garnett_test, y_garnett_pred_saga)
accuracy_LR_clf_garnett_saga
```

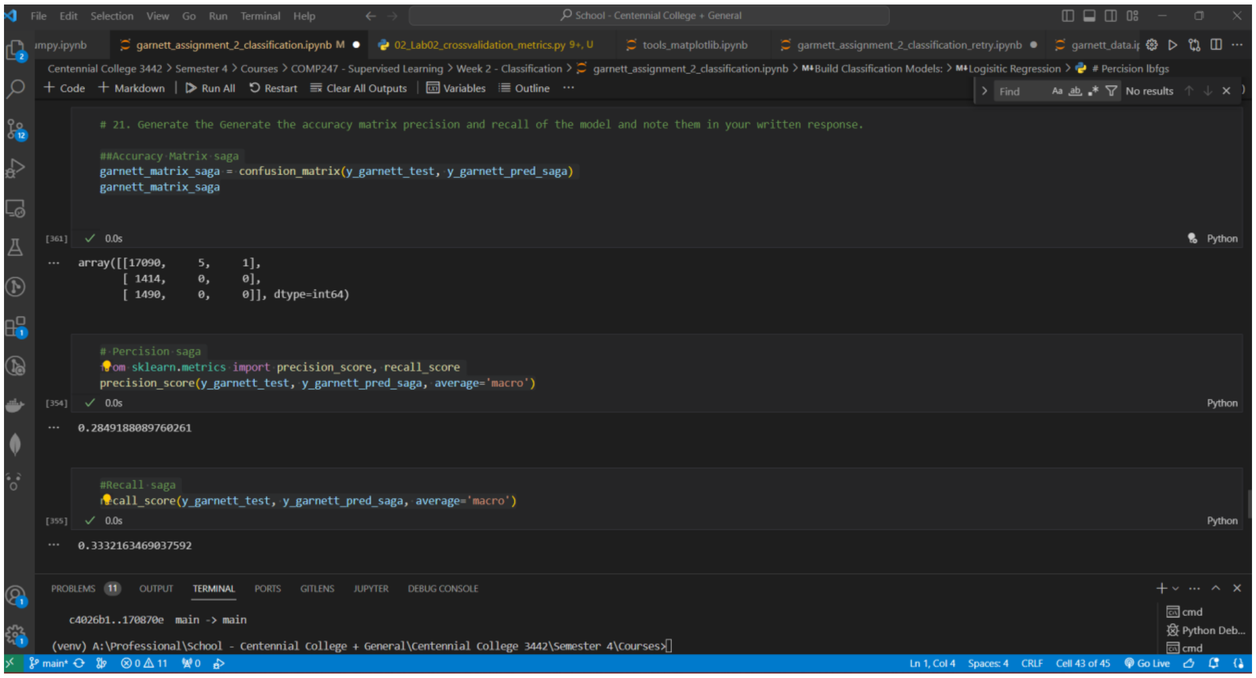
```
0.8545
```

```
# 21. Generate the Generate the accuracy matrix precision and recall of the model and note them in your written response.
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

21. Generate the Generate the accuracy matrix precision and recall of the model and note them in your written response.

The accuracy, precision, and recall of the model using both solvers are fairly similar, however, the Lbfgs solver performs a bitter better precision wise.



```
# 21. Generate the Generate the accuracy matrix precision and recall of the model and note them in your written response.

##Accuracy Matrix saga
garnett_matrix_saga = confusion_matrix(y_garnett_test, y_garnett_pred_saga)
garnett_matrix_saga
```

```
[361] ✓ 0.0s
array([[17090,     5,     1],
       [ 1414,     0,     0],
       [ 1490,     0,     0]], dtype=int64)
```

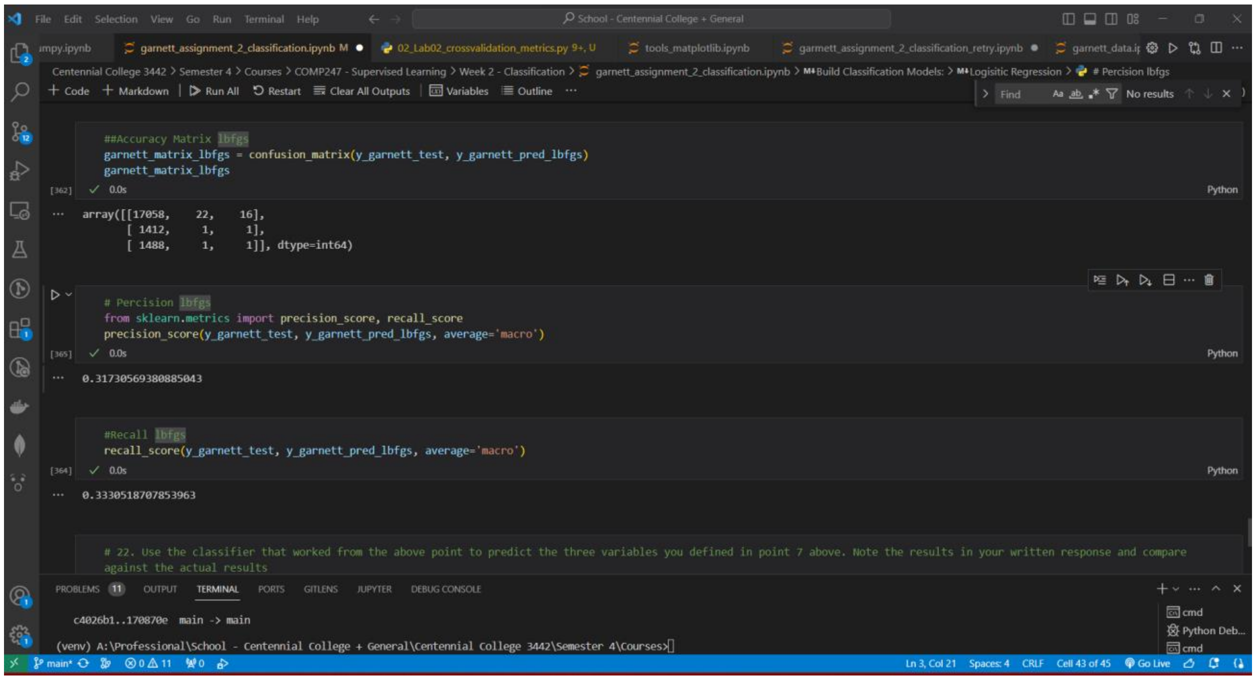
```
# Percision saga
from sklearn.metrics import precision_score, recall_score
precision_score(y_garnett_test, y_garnett_pred_saga, average='macro')
```

```
[354] ✓ 0.0s
0.2849188089760261
```

```
#Recall saga
recall_score(y_garnett_test, y_garnett_pred_saga, average='macro')
```

```
[355] ✓ 0.0s
0.3332163469037592
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```



```
##Accuracy Matrix lbfgs
garnett_matrix_lbfgs = confusion_matrix(y_garnett_test, y_garnett_pred_lbfgs)
garnett_matrix_lbfgs
```

```
[362] ✓ 0.0s
array([[17058,    22,    16],
       [ 1412,     1,     1],
       [ 1488,     1,     1]], dtype=int64)
```

```
# Percision lbfgs
from sklearn.metrics import precision_score, recall_score
precision_score(y_garnett_test, y_garnett_pred_lbfgs, average='macro')
```

```
[365] ✓ 0.0s
0.31730569380885043
```

```
#Recall lbfgs
recall_score(y_garnett_test, y_garnett_pred_lbfgs, average='macro')
```

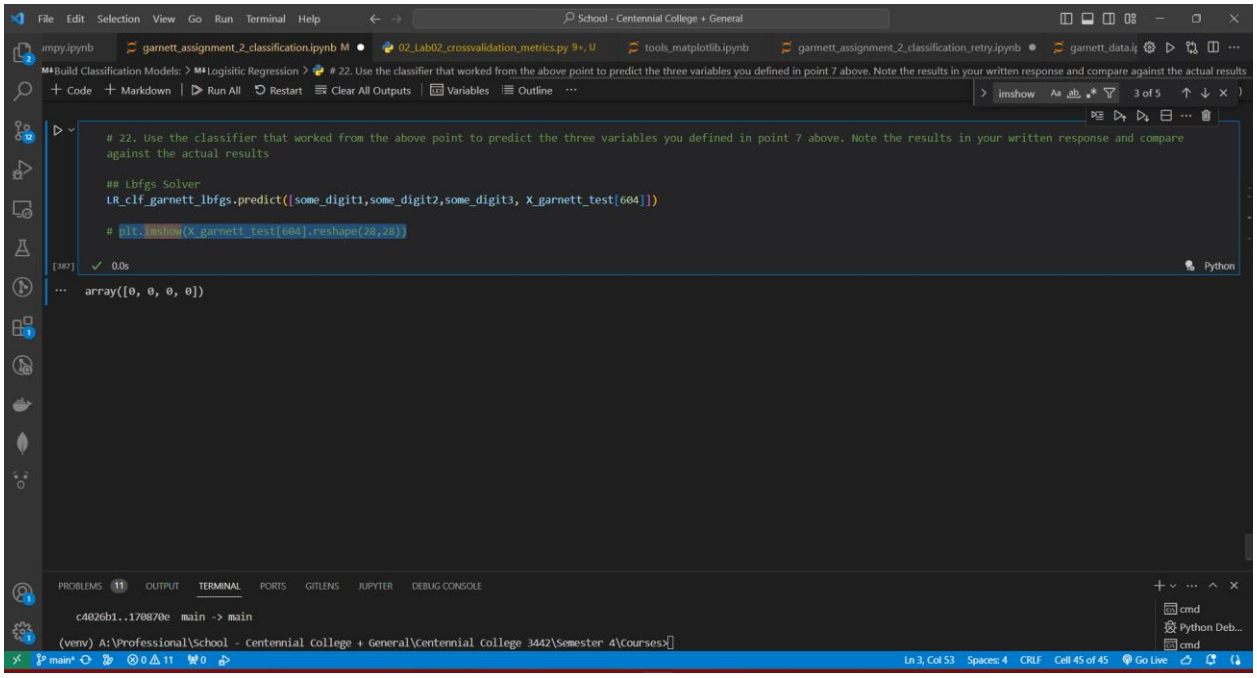
```
[364] ✓ 0.0s
0.3330518707853963
```

```
# 22. Use the classifier that worked from the above point to predict the three variables you defined in point 7 above. Note the results in your written response and compare against the actual results
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

22. Use the classifier that worked from the above point to predict the three variables you defined in point 7 above. Note the results in your written response and compare against the actual results

Using lbfgs solver:

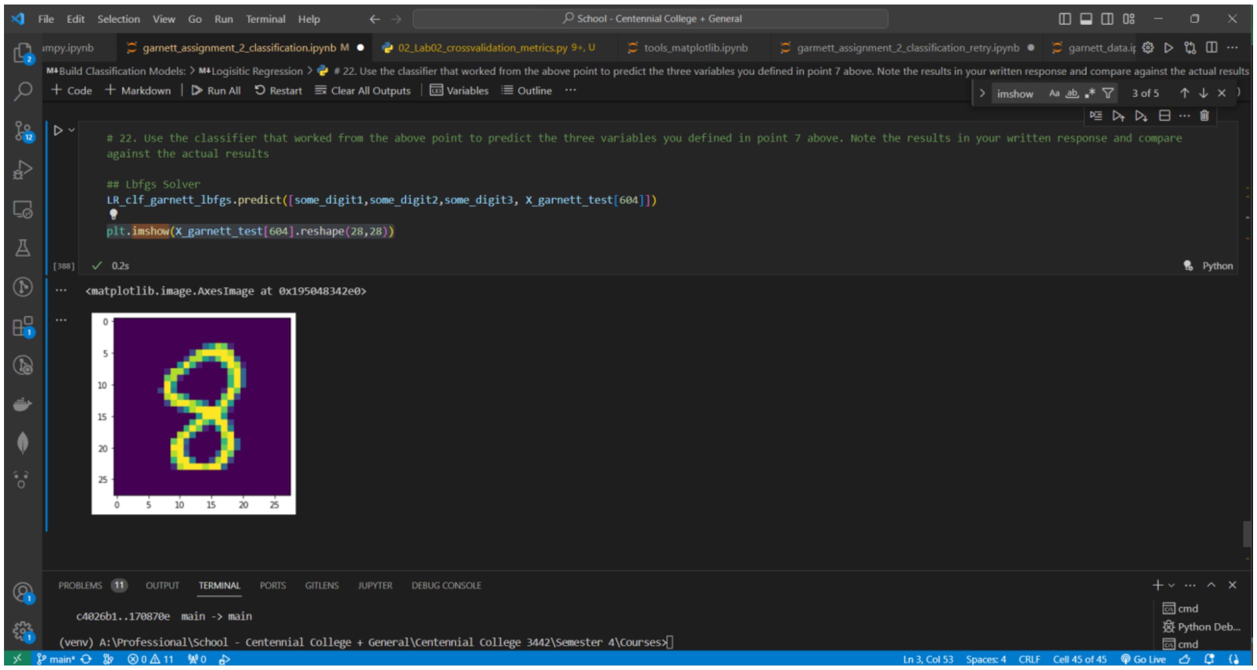


```
# 22. Use the classifier that worked from the above point to predict the three variables you defined in point 7 above. Note the results in your written response and compare
against the actual results

## Lbfgs Solver
LR_clf_garnett_lbfgs.predict([some_digit1,some_digit2,some_digit3, X_garnett_test[604]])

# plt.imshow(X_garnett_test[604].reshape(28,28))
```

```
array([0, 0, 0, 0])
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

Actual Result:



```
# 22. Use the classifier that worked from the above point to predict the three variables you defined in point 7 above. Note the results in your written response and compare
against the actual results

## Lbfgs Solver
LR_clf_garnett_lbfgs.predict([some_digit1,some_digit2,some_digit3, X_garnett_test[604]])

plt.imshow(X_garnett_test[604].reshape(28,28))
```

```
<matplotlib.image.AxesImage at 0x195048342e0>
```

```
c4026b1..170870e  main -> main

(venv) A:\Professional\School - Centennial College + General\Centennial College 3442\Semester 4\Courses>
```

Finally, in your analysis report (written response) Compare the results from both models, investigate why a model performed better than the other, and write your conclusions. (Be thorough)

When it comes to performance, the saga solver Logistic Regression model outperformed every other model.

When it comes to precision, the lbfgs solver Logistic Regression model, outperformed every other model.

When it comes to accuracy, both Logistic Regression Models utilizing the saga and lbfgs solvers, outperformed the Naïve Bayse Model.

When it comes to convergence, lbfgs continues to fail at converging in every aspect.

When it comes to prediction, all Models perform poorly.

In conclusion, my models are correctly processing the data it is given, in optimal timing when fine-tuned appropriately. There may be some other inconsistencies that need to be addressed in order for the Model to operate at it's best capacity to provide the most thorough results which would further allow the model to be used at it's greatest capacity in a real world situation.