Name (netid): Chaoyang Wang (cw39)
CS 441 - HW1: Intro to Classification and Regression

Complete the claimed points and sections below.

Total Points Claimed **[ ] / 160**

1. MNIST Classification
    a. Main Results
        i. KNN [10] / 10
        ii. Naive Bayes [10] / 10
        iii. Logistic Regression [10] / 10
        iv. Confusion Analysis [10] / 10
    b. comparison over data size [10] / 10
    c. parameter selection [10] / 10
2. Temperature Regression
    a. Main Results
        i. KNN [10] / 10
        ii. Naive Bayes [10] / 10
        iii. Linear Regression [10] / 10
    b. Most Important Features [10] / 10
3. Stretch Goals
    a. Improve MNIST classification [5] / 20
    b. Improve Temperature regression [10] / 20
    c. Create dataset where NB beats NN/LR [ ] / 20

1. MNIST

a. Main Results

	KNN	Naive Bayes	Logistic Regression
Val Error	0.0304	0.1634	0.0753
Training Time (s)	0	0.33	7.58
Inference Time (ms)	218.44	0.12	0.0016

Using your confusion matrix for KNN, report which label is most commonly confused with each true label. For example, if the most common mistake for “3” images is to assign them to “8”, then put “8” in the second row under “3” and the percent of “3”s that are classified as “8” in the third row.

	0	1	2	3	4	5	6	7	8	9
Most common mistaken label	**6**	**7**	**7**	**5**	**9**	**3**	**0, 1**	**9**	**1, 3**	**7**
% of times assigned to that label	**0.30**	**0.18**	**1.31**	**1.74**	**2.86**	**1.62**	**0.39**	**0.84**	**1.27**	**1.12**

b. Plot Validation Error vs. Training Size

# training samples	KNN	Naive Bayes	Logistic Regression
50	0.3372	0.4749	0.3503
500	0.1611	0.2393	0.1534
5,000	0.0682	0.1695	0.1085
50,000	0.0304	0.1634	0.0753

c. Parameter selection

	KNN (K)	Naive Bayes (alpha)	Logistic Regression (C)
Best parameter	1	0.1	0.7
Validation error	0.1611	0.2163	0.1531
Test error	0.1622	0.2023	0.1481

2. Temperature Prediction

a. Main Results

	KNN	Naive Bayes	Linear Regression
RMS Error	3.23	3.78	2.44
Median Abs Error	2.08	2.46	1.63

b. Most Important Features

Feature Rank	Feature number	City	Day
1	334	Chicago	-1
2	347	Minneapolis	-1
3	405	Grand Rapids	-1
4	366	Kansas City	-1
5	361	Cleveland	-1
6	307	Omaha	-2
7	367	Indianapolis	-1
8	264	Minneapolis	-2
9	9	Boston	-5
10	236	Springfield	-3

Using only the 10 most important features

	KNN	Naive Bayes	Linear Regression
RMS Error	2.82	2.79	2.26
Median Abs Error	1.57	1.51	1.24

3. Stretch Goals

a. Improve MNIST classification performance

Report the classification val and test errors and details of your best method. Describe your approach and parameters.

Effort:
1. Data argumentation. Shift one image 1 pixel to left, right, top, bottom to create 4 new images. It makes train size 250000 and improve result of methods need large dataset such as LR.
2. Try to adjust parameters of sklearn.LR including solver{‘lbfgs’, ‘sag’, ‘saga’}, C{0.1-1}, tol, multi_class, penalty{l1, l2}.

It seems that ‘sag’ outperforms in argumented large dataset and decrease helps strengthen the regularization and improve results.

Original val and test error: 0.0753, 0.0737
Improved val and test error: 0.0748, 0.0719
LogisticRegression(C=0.4, multi_class='multinomial', penalty='l2', solver='sag')

```
# test
y_pred = clf.predict(x_test)
print(f"Validation error: {np.sum(y_pred != y_test) / y_pred.shape[0]}")
```

```
Validation error: 0.0719
```

b. Improve Temperature regression performance

Report the RMS val and test errors and details of your best method. Describe your approach and parameters.

1. Do feature analysis using LASSO to find important features. The performance is great ranging top15-30 features.
2. Adjust parameters of Lasso and Ridge such as alpha and solver.

Finally I use Lasso(alpha=0.75).fit(x_train, y_train) to find most important top16 features. And use Ridge(alpha=0.6, solver='svd') to fit the model.

Final results:

```
clf = Ridge(alpha=0.6, solver='svd').fit(x_train_filter, y_train)
y_pred = clf.predict(x_val_filter)
print("Val: ")
print(f"RMSE: {np.sqrt(np.mean((y_pred-y_val)**2))}")
# print(f"MAE: {np.median(np.abs(y_pred-y_val))}")
y_pred = clf.predict(x_test_filter)
print("Test: ")
print(f"RMSE: {np.sqrt(np.mean((y_pred-y_test)**2))}")
# print(f"MAE: {np.median(np.abs(y_pred-y_test))}")
```

```
Val:
RMSE: 2.2531446688322156
Test:
RMSE: 2.049356352970692
```

c. Generate a train/test classification dataset in which Naive Bayes outperforms 1-NN and Logistic Regression

Explain how you generated the data and report test performance for each method.

Acknowledgments / Attribution

https://towardsdatascience.com/improving-accuracy-on-mnist-using-data-augmentation-b5c38eb5a903