Practical task 13.

This task consists of two big parts: one of them is about pictures, and the other one is about texts. The part about texts is optional and can give you extra points.

1 Features for images

To complete this task you are asked to implement the following functions:

```
nmi(x, y):
(nmi_value)
```

This function calculates Normalized Mutual Information of features `x` and `y`.
The arguments:

1. `x`, `y` - numpy array of shape N, storing feature values.

The return values:

1. `nmi_value` - scalar - NMI value.

```
PCA_fit(X, num_components):
(G, exp_var, X_mean, X_std)
```

This function trains PCA via SVD.
The arguments:

1. `X` - numpy array of shape $N \times D$ with data PCA trains on
2. `num_components` - scalar, the number of components in PCA

The return values:

1. `G` - numpy array of shape $D \times$ `num_components`, storing transformation matrix from initial feature space to the PC feature space.
2. `exp_var` - numpy array of shape `num_components`, storing explained variance of each PC.
3. `X_mean` - numpy array of shape D, storing mean values of the initial features.
4. `X_std` - numpy array of shape D, storing std values of the initial features.

Useful functions and classes: `numpy.linalg.svd()`

```
PCA_transform(X, center, scale, G):
(PC)
```

This function projects features of X onto PC space
The arguments :

1. `X` - numpy array of shape $N \times D$ with data to be transformed.
2. `center` - numpy array of shape D, that is used to centralize features of X.

3. `scale` - numpy array of shape D, that is used to scale features of X.

4. `G` - numpy array of shape $D \times$ `num_components`, storing transformation matrix from initial feature space to the PC feature space.

The return values:

1. `PC` - numpy array of shape $D \times$ `num_components`, calculated PC values.

Tasks:

During this set of tasks we would elaborate on feature selection techniques.
You are allowed to use `sklearn.linear_model.LogisticRegression` as logistic regression implementation and `sklearn.ensemble.RandomForestClassifier` as Random Forest implementation.

We would try to recognize a digit based on the way it is handwritten. That means that our modelling task is classification with multiple classes (particularly, 10 classes - one for each digit $0 - 9$).

1. Choose arbitrary `random_state` and use it in train-test splitting and model training procedures.

2. Load the dataset with command `digits = sklearn.datasets.load_digits()`. This dataset contains 8×8 images of handwritten digits. Labels (classes) and images (features) can be acquired via commands `digits.target` and `digits.images`. You can visualize handwritten digit
with command `plt.imshow(image, interpolation='none', cmap=plt.cm.Greys)`
You should:

   (a) Reshape image data with shape `(1797, 8, 8)` to 2-D image-feature matrix with shape `(1797, 64)`

   (b) Split the dataset into train/test sets in proportion $60/40$ respectively.

3. Train simple logistic regression (no regularization) and calculate the accuracy of classification on the test set. Refer to this result as **baseline**.

4. Selection with NMI

   (a) Use `nmi` function to calculate Normalized Mutual Information of image labels and features on train set. Illustrate your findings.

   (b) Iterate over threshold $\theta \in [0, max(\texttt{nmi})]$ with step 0.02. On each step train simple logistic regression (no regularization) using features with NMI $\geq \theta$, display corresponding number of features, accuracy of classification on the test set and compare it with **baseline**.

5. Selection with L1 regularization

   (a) Train logistic regression with L1 regularization on initial feature set. Configure regularization parameter `C=0.2`. Calculate the accuracy of classification on the test set and compare it with **baseline**.

   (b) Identify how many features have been selected in each of 10 discriminant functions.

6. Feature importances with trees

   (a) Train random forest (RF) with default settings and calculate the accuracy of classification on the test set.

   (b) Compare RF feature importances with NMI. Are they correlated?

7. PCA feature reduction

   (a) Use function `PCA_fit` on train sample to learn PCA with 60 components. Note that you MUST use SVD in this function. Plot the ratio of explained variance of each component.

   (b) Use function `PCA_transform` on test sample to transform it to Principal Component feature space. How many components are enough to acquire same or greater accuracy (comparing to the **baseline**) on the test set with simple logistic regression (no regularization)?

2 Features for texts

To complete this task you're asked to implement the following functions:

```
extract_features(train_texts, test_texts, ngrams_count):
(train_features, test_features)
```

This function extracts word n-grams (n-gram consists of n words) from raw texts and applies TF-IDF transform to review-ngram matrix. This function returns feature matrices for train and test sets.

The arguments:

1. `train_texts` - numpy array of shape N, storing train reviews, one string per review text.
2. `test_texts` - numpy array of shape M, storing test reviews, one string per review text.
3. `ngrams_count` - maximum size of n-gram features (example: for `ngrams_count` = 2 resulting features must contain unigrams and bigrams)

The return values:

1. `train_features` - feature matrix for train set of size $N \times D$
2. `test_features` - feature matrix for test set of size $M \times D$

Useful functions and classes: `sklearn.feature_extraction.text.TfidfVectorizer`

```
logistic_regression(train_texts, train_labels,
test_texts, test_labels, ngrams_count, penalty, minC, maxC, steps)
```

This function extracts features from raw texts (see `extract_features` above) and trains logistic regression with specified regularizer `penalty` ("l1"or "l2") and regularization parameter C found by 5-fold cross-validation on training set. The accuracy of resulting classifier on test set is printed alongside the number of features with non-zero weights.

The arguments:

1. Look at the arguments of `extract_features`.
2. `test_labels` - test labels.
3. `train_labels` - train labels.
4. `penalty` - regularization type ("l1"or "l2").
5. `minC`, `maxC` - minimal and maximal values of C, defining the range for figures.
6. `steps` - number of points in generated range of C values.

Useful functions and classes: numpy.logspace, `extract_features`, `sklearn.grid_search.GridSearchCV`.

The task:

1. You're given a dataset of IMDB movie reviews (`https://www.dropbox.com/s/rz024e4p71dd05j/imdb.zip?dl=0`), containing texts for positive/negative sentiment classification. Each review is stored as a seperate text file in folders named `train_pos`, `train_neg` for 2 classes in training set and `test_pos`, `test_neg` for test set. Firstly, load train and test sets in memory as arrays of strings, where each review is reperesented by one string.
2. Using the function `logistic_regression`, analyze the accuracy of "l1"and "l2"regularized logistic regressions with 1-grams (`ngrams_count` = 1) and 2-grams (`ngrams_count` = 2). The values of C must be in the logarithmic scale in the range of $[10^{-8}, 10^{8}]$ with at least 17 steps.
   (a) Does addition of bi-grams lead to better performance?
   (b) Which penalty yields better accuracy on test set?
   (c) When does it make sense to use "l1"penalty?

3. From now on we will use ngrams_count $= 2$. Train "l1"regularized logistic regression with big enough C parameter to obtain roughly 20% of features with non-zero weights. On resulting features train "l2"regularized logistic regression with cross-validation. Obtain test accuracy score. Compare with other results from previous steps. What conclusions can you make?

4. Use sklearn.linear_model.RandomizedLogisticRegression to train many "l1"regularized logistic regressions with parameter C from previous step. After calling "fit"on training, the model will contain feature importances in "scores_"field. Take 5% of features with highest importance and train "l2"regularized logistic regression with cross-validation. Obtain test accuracy score. Compare with other results from previous steps. What conclusions can you make?