Homework #4

Objective:

This homework's objective is to train a Support Vector Machine (SVM) model to predict the tissue of origin for cancer samples. This is a bioinformatics project that introduces you to using RNA data for prediction tasks.

Details:

Ribonucleic acid (RNA) is a type of molecule produced by living cells from templates encoded by the deoxyribonucleic acid (DNA) molecules. DNA contains the blueprint for RNA molecules, of which messenger RNA (mRNA) is the most well-known type of RNA molecule. Each gene can be a blueprint for one or more variants of an mRNA molecule representing that gene. Proteins, the building blocks of life, are produced by living cells from the information encoded by mRNA molecules. Understanding how much of each protein a cell produces can tell us a lot about the cell characteristics and behavior. Quantifications of the amount of mRNA molecules in the cell can be used as a proxy for the amount of proteins in that cell. The more of a particular mRNA we detect, the more of the protein encoded by the mRNA we expect the cell to produce. Current sequencing technologies allow sequencing both DNA and RNA molecules in a timely and cost-efficient manner. Protein quantifications are not as easy to obtain. Therefore, sequenced RNA is often used to approximate protein levels. Different types of cells expression different amounts of various mRNA molecules. In fact, mRNA expression profile of a cell is predictive of the tissue of origin for that cell.

In this assignment you are going to use mRNA expression data for cancer samples from various cancer types, collected as a part of international multi-institutional long-running collaboration of researchers and clinical professionals called The Cancer Genome Atlas (TCGA). Over the years various important projects and publications came out of this initiative, which greatly contributed to the field of cancer genomics research. You can find more information about the TCGA project and publications related to it at the official site:
https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga

The dataset you will be working with comes from a 2013 publication in Nature Genetics titled "The Cancer Genome Atlas Pan-Cancer analysis project":
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3919969/
In this publication the authors described a joint analysis of genomic data from 12 different types of cancer types (by the tissue of origin).

I already downloaded and wrangled the TCGA gene expression data into a csv file for you to use. The rows of this file are cancer samples and the columns are mRNA expression quantifications for individual genes. The last column of this file is a cancer type assignment (class). This label is the tissue of origin for each of the cancer samples and is the output variable to be predicted. Your assignment is to build a predictive model for this label from the other variables, input variables. You are going to use an SVM algorithm to train this model.

As some of you might already know there are ~20,000 protein coding genes in human genome. For this assignment I already performed feature extraction in order to get the input file to a more manageable size. I extracted 3,000 most varying features for you to train your model. Often in genomic research we use most varying features for prediction tasks. That means that we want to select those features that have the highest variance across the training dataset. For this type of dimensionality reduction we compute variance for each feature across all training samples (each column) and select those 3,000 genes/features that have the highest variance.

Use input file homework4_input_data.csv for this assignment.

Notice that there are more than two tissues of origin labels in this dataset (12 cancer types). This means that you are going to perform a multi-class prediction task.

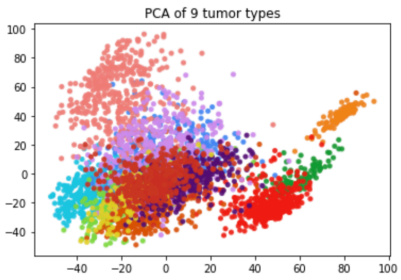
RNA sequencing data is known to be exponentially distributed. This means that if you plot gene expression values for individual samples you will see that they are distributed according an an exponential distribution (most genes are expressed at low levels while fewer genes are highly expressed). This is because only a small subset of genes is usually activated in any particular tissue and these subsets vary across different tissues. As you probably remember from the lecture material, models such as SVM best learn on data that is normally distributed (according to a Gaussian distribution). Therefore, you will need to use the StandardScaler to normalize the data. Refer to my example notebooks to see how to rescale the data or use internet. If you use code examples from the internet then make sure to site your sources in your notebook. Use the following import statement for the StandardScaler:

from sklearn.preprocessing import StandardScaler

As a part of this assignment you will need to plot a PCA plot for the input data. Please use the following colors below for your plot:

```
colors = {"Breast": '#4287f5',
        "Bladder": '#19c5e3',
        "Colon": '#80d941',
        "Glioblastoma": '#179933',
        "Head&Neck": '#f07e78',
        "Kidney": '#f01e13',
        "Leukemia": '#f0841f',
        "LungAdeno": '#db5209',
        "LungSquamous": '#ce8ced',
        "Ovarian": '#551075',
        "Rectal": '#e3d329',
        "Uterine": '#cc3423'}
```

You should be able to produce a plot similar to the example below:



For this assignment you will need to perform 5-fold cross validation. This is how model evaluation is performed in the real world. Cross-validation results are usually reported in addition to test set evaluation.

You have two options to obtain cross-validation results:

1. You can manually reak your training set into 5 stratified cross-validation sets and train and evaluate the model for each validation set. For each validation set you will need to use the rest of the training data to train the model and then score this model on the validation set (compute accuracy on this validation set). Since you will do 5-fold cross validation, you will have 5 accuracies across 5-folds. Store them as you go and report individual accuracies as well as the mean 5-fold cross-validation accuracy at the end. You can assess the model performance for each fold by simply computing the over prediction accuracy using *score()* method. As you do this, store the accuracies and print them out after you have gone through all 5 folds, as well as compute and print the mean accuracy across the 5 folds. Use *StratifiedKFold* to break your input data into 5 folds:

   from sklearn.model_selection import StratifiedKFold

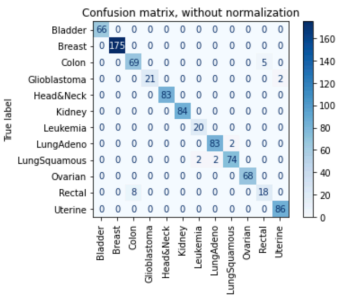
2. Alternatively, you can use *cross_val_score()* method available in *sklearn.model_selection.* This method already utilizes stratified cross validation when the predictor variable is a class label. Import the appropriate library as follows:

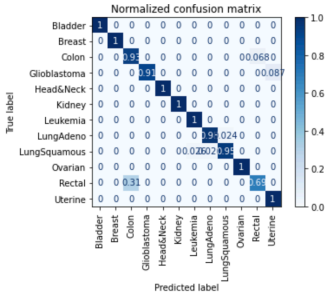
from sklearn.model_selection import cross_val_score

Reporting/printing out something like this for your 5-fold cross-validation results is sufficient for this assignment:

```
Individual cross-validation accuracies: [0.9610951008645533, 0.9582132564841499, 0.968299711815562,
0.9711399711399712, 0.9624819624819625]
Mean cross validation accuracy 0.96
```

After performing 5-fold cross-validation, train the final model on all of the training data and compute the final accuracy using test set. In addition to reporting test set accuracy plot two confusion matrices, one non-normalized and one normalized (refer to my examples for how to do that). Your confusion matrices should look something similar to these (your values may not be exactly like mine because your model hyperparameters might differ):





To train an SVM you can use several existing scikitlearn libraries. You are free to use any of these libraries for this assignment.

✶ sklearn.svm.LinearSVC - this implementation of SVM performs one-vs-all classification tasks for every class in the multi-class dataset. Use these parameters for your SVM model: multi_class='ovr', class_weight='balanced'.

✶ sklearn.svm.SVC - this implementation of SVM performs one-vs-one classification tasks for every pair of classes in the multi-class dataset. I did not show examples of using this model in class. You are responsible for figuring out how to use it if you choose to do so. If you use code examples from the internet then make sure to site your sources in your notebook.

You can use the code from my notebook examples as a reference to help you get started:

- SVM.Breast.ipynb
- SVM.Iris.ipynb

<u>Your submission should include the following:</u>

1. Load the data.
2. Produce a PCA plot of the input data, using the colors specified above.
3. Normalize the data using StandardScaler.

4. Break the data into the training and test datasets at 80/20 proportion. Make sure that you stratify this break-up as there are 12 classes and you want to avoid test data that is class-imbalanced and does not accurately represent your training data. Notice that train_test_split method has an input parameter called *stratify*. You will need to specify this parameter to be value "Y".
5. Define SVM model hyperparameters of your choice (e.g. kernel, regularization, etc.). These are parameters into the model initialization.
6. Run and report results from 5-fold cross-validation. Use one of the methods I outlined in the description above.
7. Train the final model on all the training data and assess model performance on the test set. This means that you need to compute prediction accuracy on the test set.
8. Plot two confusion matrices for test set predictions (one non-normalized and one normalized). You can choose to use the same implementation of plotting a confusion matrices as I showed in my examples or include a different implementation. If you use code examples from the internet then make sure to site your sources in your notebook.