14:00 – 14:00hrs	6 – 10/Dec/2021		EXAMS OFFICE USE ONLY

University of the Witwatersrand, Johannesburg

Course or topic No(s)	COMS4060A/7056A
Course or topic name(s) Paper Number & title	Intro to Data Visualisation and Exploration
Examination to be held during the month(s) of	December, 2021
Year of study	
Degrees/Diplomas for which this course is prescribed	
Faculties presenting candidates	Science
Internal examiner(s)	Dr Timothy Bristow
External examiner(s)	Dr Paresh Malalur
Special materials	Dataset Provided on Moodle
Time allowance	72 Hours
Instructions to candidates	125 Marks available. 125 marks = 100%. Answer all questions. This is an **open-book** exam: you may consult any material except do **not** use Kaggle. This exam consists of 9 pages and 3 questions. Submit a Jupyter notebook, or PDF with source code, on Moodle by 14:00 10/12/2021. Hard deadline 17:00, penalties will apply.

Introduction to Data Visualisation and Exploration
Final Exam

Datasets and Code

You can find all of the *required* datasets available for download on Moodle in `datasets.zip`. You can augment this with other datasets you find online, but you must provide a link to where you found the data, and consider the trustworthiness of your sources.

You can find and use code from the Jupyter notebooks provided and linked throughout the course for performing many parts (but not all) of this exam.

Question 1 Netflix [43 Marks]

Dataset Overview

You are provided with a dataset, `netflix_titles.csv`, containing TV shows and movies available on Netflix US, which you will need to augment with IMDB data. Perform exploratory data analysis and provide visualisations of this combined dataset to answer questions based on the data.

The Netflix data contains the following:

- `title`: Title of the movie or TV series.
- `type`: Movie or TV show.
- `duration`: Duration in minutes for movies, or number of seasons for TV shows.
- `director`: Director of the movie or show.
- `cast`: Leading cast members.
- `rating`: The age/maturity rating of the content.
- `country`: Country of *origin* of the content.
- `release_year`: Year that the content was released (not necessarily on Netflix).
- `date_added`: Date that the content was made available on Netflix US.

The file `title.basics_small.tsv.gz` is from IMDB, and contains the title of the movie, TV show, or episode of a TV show, the release dates, genres, duration, and a corresponding titleId. Using titleId you can join this to the other IMDB files on `titleId/tconst`.

The other IMDB datasets include:

- `title.akas_small.tsv.gz`: localisation information for the content, language, region,
- `title.crew_small.tsv.gz`: directors and writers,
- `title.episode_small.tsv.gz`: episode and season number ,
- `title.principals_small.tsv.gz`: cast and crew,
- `title.ratings_small.tsv.gz`: user ratings and the number of votes,
- `name.basics_small.tsv.gz`: names of actors and directors in the above files (join to above datasets on `nconst`).

Note: these files are *tab-separated* files. You can read these files in pandas using the command

```
pd.read_csv('title.basics.tsv.gz', delimiter='\t')
```

The complete IMDB dataset description is provided in `imdb_description.pdf`, alternatively, you can access it here https://www.imdb.com/interfaces/. Note that the files you are given for the exam are subsets of those available on the website: the original datasets are *large*.

1.1 Exploration of Netflix data

Start with the Netflix dataset (not yet IMDB):

1. Describe the dataset: number of entries, range/mean of the variables, number of unique movies, for example. [3]
2. Identify missing values and duplicates. How many missing values are there per column? [3]

For the below, show **plots** where required and **always interpret** the data.

4. Provide a graph showing the split between movies and TV shows. [2]
5. Show the distribution of movie duration (or *number* of seasons of TV shows). [3]
6. Plot the age of content when it is released on Netflix. If you were tasked with finding Netflix *produced* content, how would you identify it here (without IMDB data)? [3]
7. How is it distributed by age/maturity rating? [2]
8. How is it distributed by genre? Which genres are the most popular? [2]
9. Are TV shows or movies added more regularly? Provide a line graph showing how this changes over time. [2]
10. Is there a particular time of week/year when content gets uploaded? Would you suggest there is a trend in this series? [3]

1.2 IMDB Ratings

The Netflix dataset does not include ratings for the movies or shows. Whilst Netflix provides information about the content being added, it is not clear if this is actually *popular* content. There are a few ways this could be handled: find Netflix ratings, or, estimate this by getting data from other sources like IMDB or Rotten Tomatoes. Here you will be using IMDB. For this section, only consider **movies** in the Netflix dataset.

1. Join the Netflix dataset to `title.basics.tsv.gz` and `title.ratings.tsv.gz` to find a rating (out of 10) for each movie. Do your results make sense? [3]
2. When augmenting with IMDB data, consider that some titles will have a differences in the name (a missing "The", for example), so not all will match when doing a naive join - what would you do with these missing joins? Attempt a strategy to improve the number of matches by 5%. [3]
3. Plot the ratings of the movies. Does Netflix have a good *quality* library? [2]
4. Investigate the relationship between the movie and its duration, director, cast, genre. You can show these as plots, a heatmap, or show the correlation. Do you find any trends? Is the most popular genre the highest rated genre? [6]

1.3 Actors and Directors

In the initial exploration of the dataset, you should have noticed missing data for actors and directors: some can be filled in using data from IMDB. Only consider **movies**.

1. How many missing directors are you able to identify? [2]
2. Which director has the most titles before filling in the missing data? Do they still have the most titles after filling in the missing data? [2]
3. Does this director (augmented with IMDB) get the best movie ratings of all directors (consider only using directors that have appeared more than 3 times)? [2]

Question 2 Human Activity Recognition [67 Marks]

Dataset Overview

Activity recognition using smartphones (or smart devices more generally) has many use cases and potential. Examples include sports tracking, or monitoring elderly or less-abled people. This is a difficult problem to solve for several reasons: people have different movements, devices have different components and capabilities (different phones, watches, etc), the implementation might change (Android vs iOS), and the design and implementation of algorithms that can identify the activities.

A well-known dataset (which was discussed in the lectures) is the Human Activity Recognition dataset from D. Anguita et. al. See here for details https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones.

Description of the dataset from the authors:

> The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz.
>
> The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

A complete description of the data is given in the attached file: `UCI HAR Dataset.names`.

For each record in the dataset the following measurements are provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables (many of which are derived from the accelerometer and gyroscope readings, see below).
- Its activity label.
- An identifier of the subject who carried out the experiment.
- Note that each window is 1.28 seconds long.

The following terms might be useful in deciphering the data (these are included for most of the signals):

- `mean()`: Mean value
- `std()`: Standard deviation
- `mad()`: Median absolute deviation
- `max()`: Largest value in array
- `min()`: Smallest value in array
- `sma()`: Signal magnitude area
- `energy()`: Energy measure. Sum of the squares divided by the number of values.
- `iqr()`: Interquartile range

- `entropy()`: Signal entropy
- `arCoeff()`: Autoregression coefficients with Burg order equal to 4
- `correlation()`: correlation coefficient between two signals
- `maxInds()`: index of the frequency component with largest magnitude
- `meanFreq()`: Weighted average of the frequency components to obtain a mean frequency
- `skewness()`: skewness of the frequency domain signal
- `kurtosis()`: kurtosis of the frequency domain signal
- `bandsEnergy()`: Energy of a frequency interval within the 64 bins of the FFT of each window.
- `angle()`: Angle between two vectors.

2.1 Exploration

When working with this dataset, the typical goal is to classify activity into one of the labels: walking, lying down, etc. Here, the goal is exploratory data analysis, which is done *before* the classification step. Perform your exploration on the *training* dataset.

1. Describe the dataset: number of entries, range/mean of the variables. [1]
2. Are there any missing values or duplicates? [1]
3. How many of each class? How many for each user? Provide basic visualisations for this. [3]

Broadly, these activities can be divided into dynamic and static movements (walking vs sitting or lying down, for example).

4. Plot some of the accelerometer readings for all classes and find one that shows a clear static/dynamic separation. Include a box-plot to discuss and convey this information clearly. [3]
5. Identify a simple rule (if/else, basically) on this accelerometer reading that can be used to separate dynamic from static movements. [1]

Intuitively, one would expect that lying down would have minimal movement, but have an identifiable angle between the phone sensor axes and the direction of gravity.

6. Identify and plot a feature that exploits this, showing class separation. [2]
7. Give a simple rule that effectively separates LAYING from all other classes. [1]

It should be possible to separate some of the classes with some straightforward rules:

8. Use t-SNE to project the data, coloured by class. Comment on the results. [3]

Another question we might ask is if we can identify individual *users* based on data for specific activities.

9. Plot a t-SNE projection, colouring the labels by the user id. Comment on the results. [2]
10. Investigate WALKING. Plot the difference in walking for different users: step frequency or lateral movement, for example. You choose the features. [4]
11. Plot a bar chart showing the difference in walking speed (or duration) going up or down stairs for all users (ie, all users in a single plot). [3]
12. Given your findings, comment on the feasibility of identifying users from walking data using simple decision rules. [2]

2.2 Baseline models

In the above question, you should have identified that without applying any sophisticated modeling you are already able to get insights and draw valuable information from the data. We will now start looking at some baseline models to get more out of the data.

Define three baseline models (with reasonable parameters) for this section (all of these are in sklearn, and you have seen RFC and logistic regression in the example notebooks during class):

- Random Forest Classifier
- Logistic Regression
- Support Vector Classification with the radial basis function ("rbf") kernel

When setting up the models, always set `random_state=`**`42`**. [1]

Use F_1 score to measure the performance of your classification. Remember, $F_1 = 2\frac{precision \cdot recall}{precision + recall}$, where *Recall = True Positives / (True Positives + False Negatives)* and *Precision = True Positives / (True Positives + False Positives)*. However, you can use the built-in functions in `sklearn` if you are using Python.

Do not forget to test on the *test* dataset! Interpret and comment on your findings.

1. Use the three baseline models with all 561 features to predict the activity type. Record the F_1 score and the time taken to train the model. [6]
2. Use the three baseline models with all 561 features to predict the **user** for the WALKING label. Record the F_1 score and the time taken to train the model. [6]

2.3 Feature Selection

The dataset has 561 features, which likely have some correlations, and not all will be useful (maybe even harmful). Having a large number of features is going to affect the training time of the model too.

1. Perform mutual information feature selection (`mutual_info_selection` in sklearn) to select the top 5, 10, 50, 100 features. Using the baseline models, investigate how the reduction in features affects the training time and the F_1 score (you need to run each model for all feature sets). Comment on your results. [6]

2. Now, look at only WALKING and perform mutual information feature selection (`mutual_info_selection` in sklearn) to select the top 5 and then 10 features. Using the baseline models, investigate how the reduction in features affects the training time and the F_1 score. Comment on your results. [4]

2.4 Feature Extraction

In the previous section you took the existing features and found subsets of these which could be used for training. Another approach is to either engineer or extract features from the data. Perform the following dimensionality reduction techniques (you have code to perform this from the lectures):

- PCA
- Modified Locally Linear Embedding
- ISOMAP
- UMAP

You need to identify suitable parameters for each of the models: number of components, number of neighbours, etc.

1. Show a 2D plot of the embeddings/principal components, and comment on the results. [8]

2. Train the baseline models on these embeddings (your features are the output from PCA/LLE/ISOMAP/UMAP). Record the training time and the F_1 score. [6]

3. Discuss your findings on the trade-off between training time, performance, and interpretability. Compare the baseline models with no feature reduction, when using feature selection, and using the feature extraction methods here. [4]

Question 3 Self-Organising Maps [15 Marks]

Generate 200 data points with three features, lying close to a helix. Define $X_1 = \cos(s) + 0.1 \cdot Z_1$, $X_2 = \sin(s) + 0.1 \cdot Z_2$, $X_3 = s + 0.1 \cdot Z_3$ where s takes on 1000 equally spaced values between 0 and 2π, and Z_1, Z_2, Z_3 are independent and have standard Gaussian distributions.

1. Show a 3D scatter plot of the data points you have created. [3]

2. Fit a SOM to the same data, and see if you can discover the helical shape of the original point cloud. Use a 3D plot (scatter/wireframe) to show the learned weights of the SOM changed (ie, how the map is deformed after training). [5]

When training the SOM:

- Consider whether to normalise or standardise the data. [2]
- Choose a reasonable map size for the SOM (a good rule of thumb is $5 * \sqrt{N}$, where N is the number of data points). Eg. 1000 datapoints: $\approx 5 \times 30 = 150$. So, for this example, a square of size 12 x 12 should be sufficient. [3]
- Choose a suitable neighbourhood size (*sigma* parameter in minisom). [2]

Hint: for this one, try sigma values of between 1.5 and 4. You can get the map from the trained SOM in minisom using get_weights(). To get equally spaced points look at np.linspace.

There are many examples available on the minisom Github page https://github.com/JustGlowing/minisom/tree/master/examples which might be useful.