Reducing Traffic Mortality in the USA

This project lets you apply skills from:

- **Introduction to Shell**, including how to navigate the file system and view files
- **Data Manipulation with pandas**, including reading, exploring, filtering, grouping, and reshaping data
- **Merging DataFrames with pandas**, including how two merge two DataFrames
- **Unsupervised Learning in Python**, including KMeans clustering, dimensionally reduction through PCA, and visualizations using `matplotlib`
- **Supervised Learning with scikit-learn**, including multivariate regression
- **Intermediate Python for Data Science**, including visualizations using `matplotlib`
- **Intermediate Data Visualization with Seaborn**, including statistical visualizations using `seaborn`

We recommend that you review the appropriate sections of those courses before starting this project.

Here are **three charts** illustrating the road accident fatality situation described in the notebook's first paragraph.

Helpful links:

- **Manipulating Files and Directories**
- **How to run shell commands in a Jupyter Notebook** (through the underlying IPython interpreter)
- For viewing the first few lines of a file using shell commands, exercise **number 3** and **number 5** in the **Manipulating Files and Directories**

The below given are the task involved in the project. It is advised to try them before looking the solution notebook:

Task 1: Instructions

Explore your current folder and view the main dataset file.

- Check the name of the current folder using `!pwd`.
- List all files in this folder using `!ls`.
- List all files in the `datasets\` folder using `!ls` and the name of the folder.
- View the first 20 lines of `road-accidents.csv` in the `datasets\` folder using `!head`.

Task 2: Instructions

Read in the main dataset file and start exploring the data.

- Import the `pandas` module aliased as `pd`.
- Read in `road-accidents.csv` (which is in the `datasets/` folder) using `read_csv()` from `pandas`. Set the `comment` and `sep` parameters based on the output from task 1.
- Save the number of rows columns as a tuple, using the `shape` attribute.
- Generate an overview of the DataFrame using the `info()` method.

Task 3: Instructions

Create a textual and graphical overview of the data.

- Compute the summary statistics of all columns in the `car_acc` DataFrame, using the `describe()` method.
- Create a pairwise scatter plot to explore the data, using `sns.pairplot()`.

Task 4: Instructions

Explore the correlation between all column pairs in the DataFrame.

- Compute the correlation coefficient for all column pairs in `car_acc`, using the `corr()` method.

Task 5: Instructions

Fit a multivariate linear regression model using the fatal accident rate as the outcome.

- Import the `linear_model` function from `sklearn`.
- Create the features and target DataFrames, by subsetting the DataFrame `car_acc`.
- Create a linear regression object, using `linear_model.LinearRegression()`.
- Fit a multivariate linear regression model, using `fit()`.
- Retrieve the regression coefficients from the `coef_` attribute of the fitted regression object.

Task 6: Instructions

Perform a principal component analysis on the standardized data.

- Standardize and center the feature columns, using the `StandardScaler` from `sklearn` and its `fit_transform()` method.
- Import the PCA class from `sklearn`.
- Fit the standardized data to the PCA class using its `fit()` method.
- Compute the cumulative proportion of variance explained by the first two principal components, either by adding them together or by using the cumulative summation method (`cumsum`) of the explained variance array.

Task 7: Instructions

Transform the data and visualize the first two principal components in a scatter plot.

- Create a `PCA` object with two components. Assign the result to the variable, `pca`.
- Transform the scaled features using two principal components and the `fit_transform()` method of the `PCA` object.
- Extract the first and second component to use for the scatter plot. Assign the results to `p_comp1` and `p_comp2`, respectively.
- Plot the first two principal components in a scatter plot, using `plt.scatter`.

Task 8: Instructions

Cluster the states using the KMeans algorithm and visualize the explanatory power for different numbers of clusters.

- Import `KMeans` from `sklearn.cluster`.
- Initialize the `KMeans` object using the current number of clusters (k).
- Fit the scaled features to the `KMeans` object.
- Append the inertia for `km` to the list of inertias.
- Plot the results in a line plot using `matplotlib.pyplot.plot`. This type of plot is also called a scree plot.

Task 9: Instructions

Highlight the clusters of the K-means fit with three clusters in the PCA scatter plot.

- Create a `KMeans` object with 3 clusters, setting `random_state` to 8 as in the previous task.
- Fit the data to the `km` object.
- Create a scatter plot of the first two principal components and color it according to the KMeans cluster assignment.

Task 10: Instructions

Visualize the distribution of speeding, alcohol influence and percentage of first-time accidents in a direct comparison of the clusters.

- Create a new column with the labels from the KMeans clustering, using `km.labels_`.
- Reshape the DataFrame to the long format, using `pd.melt()`. Use the features as the value variables and give them the name `'measurement'` in the new DataFrame. Name the value column `'percent'`.
- Create a violin plot splitting and coloring the results according to the km-clusters using the `hue` parameter. Plot the measurements along the y-axis and the percent values along the x-axis.

Task 11: Instructions

Add data on the number of miles driven per state to compute total number of fatal accidents and total accidents for each cluster.

- Merge the `car_acc` DataFrame with the `miles_driven` DataFrame. Merge on the common column `state`.
- Create a new column for the number of drivers involved in fatal accidents. Use the columns `'drvr_fatl_col_bmiles'` and `'million_miles_annually'`, note that these are in billions and million respectively.
- Calculate the number of states in each cluster and their average and total number of drivers involved in fatal accidents using the DataFrame `agg()` method.
- Using `sns.barplot`, create a bar plot of the total number of fatal accidents per cluster, setting the `estimator` parameter to `sum`.

Task 12: Instructions

Decide which cluster to focus your resources on.

- Which cluster would you choose: 1, 2, or 3? Assign one of these integers to `cluster_num`.