COMPUTING SUBJECT:	Linear regression on 50 startups in US.
TYPE:	Mandatory project
IDENTIFICATION:	ML Startups
COPYRIGHT:	*Michael Claudius*
LEVEL:	Medium
TIME CONSUMPTION:	10-15 hours
EXTENT:	300 lines codes mainly auto-generated
OBJECTIVE:	Data Mining Linear Regression theory and application implementation
PRECONDITIONS:	See special paper with useful links
COMMANDS:	

MANDATORY PROJECT: STARTUPS

The Mission

You are to gain knowledge on machine learning by training linear regression algorithms on a specific data set. The answers will fall in two parts:

1. Theoretical part, explaining some concepts of linear regression.
2. Practical part, training and evaluating the program on a specific data set, *50_Stratups.csv*,

   *This can best be and **must** be done in small groups of 3-5 students.*

As this topic is rather green, as well as new to you, the project is defined as an exploring project; meaning that a 100% successful investigation of point 2 cannot be guaranteed, neither is it expected to be.

Purpose

The purpose of this project is to explore linear regression.

Useful links for ML

When surfing on the net it is easy to find many descriptions more or less useful, and in more or less updated versions. I have made a preliminary collection on the home page.
Furthermore, the Housing example in chapter 2 is a really good starting point !

Hand in

It is important to understand both the theory, and practical part. Therefore both parts are handed in as one .zip file not later than 23.00 14th March 2022. *For each group only one student need/have to upload the group work.*

Domain description

JPM-Finance is an advisory financial company with a small network of rich clients who like to invest in newly started companies (so called *startups*), before the profit is public. At the teachers home page you will find data on 50 startups companies in United States.
For each company are given data on the features: State, R&D Spend, Administration, Marketing Spend, and the label: Profit.
Your job is to understand, explore and prepare the data, do a linear regression analysis to be used by JPM-Finance evaluating new startups profit based on the features. The estimate of profit together with a risk analysis of the business segment, will be the foundation for financial advices to clients.
A preliminary interview with the smart boss (nick name JP) and his younger energetic coordinator, Mike, has revealed that:

a. A few companies have some data values to 0.0, this is for the moment acceptable, meaning you don't need to change these numbers into mean values or drop the respective companies. Some small startups actually don't have any administration costs!
b. The "State" feature is a text attribute, and from previous study not very important, meaning one can drop this feature. Thus, all calculations using *"import OneHotEncoder"* are superfluous. Just skip them for a start. Later if You like, You can utilize the *OneHotEncoder*.
c. If the correlation matrix of the features only has values higher than 0.20, it is a pretty bad

idea to try to combine attributes, as this can blur the picture and make wrong weights to the features.

The process

You are to follow a light weight version of the ML Management Checklist. (See the home page). That means the only some points are relevant for you. Irrelevant points are marked N/A.

Step 1: Look at the big picture and frame the problem

1. Define the objective in business terms.
2. How shall your solution be used?
3. N/A. What are the current solutions/workarounds (if any)?
4. How should you frame this problem (supervised/unsupervised, online/offline, etc.)?
5. How should performance be measured?
6. Is the performance measure aligned with the business objective?
7. N/A. What would be the minimum performance needed to reach the business objective?
8. What are comparable problems? Can you reuse experience (of course YES) or tools?
9. N/A. Is human expertise available?
10. N/A. How would you solve the problem manually?
11. *List the assumptions you (or others) have made so far.*

Step 2. Get the data

The data is the 50_Startups.csv file at the teachers home page.

1. N/A. List the data you need and how much you need.

   N/A. Find and document where you can get that data.
2. Check out how much space it requires. (Well its super small ☺)
3. N/A. Check legal obligations, and get authorization if necessary.
4. N/A. Get access authorizations.
5. Create a workspace (with enough storage space) at your PC for the program and data set,
6. Get the data.
7. N/A. Convert the data to a format you can easily manipulate (without changing the data itself). It's already good, already.
8. N/A. Ensure sensitive information is deleted or protected (e.g., anonymized).

9. Check the size and type of data (time series, sample, geographical, etc.).
   Just use "Notepad".
10. *Document what you have learned.*

Step 3. Explore the data

You are now ready to make a Jupyter program for exploring the data.
Remember your experience from the investigation of "Housing" in Chapter 2..

1. Create a copy of the data for exploration (N/A. sampling it down to a manageable size if necessary).
2. Create a Jupyter notebook to keep a record of your data exploration.
   A copy and paste and changes of the *Housing* is not illegal ☺
3. Study each attribute and its characteristics:
   - Name
   - Type (categorical, int/float, bounded/unbounded, text, structured, etc.)
   - % of missing values
   - N/A. Noisiness and type of noise (stochastic, outliers, rounding errors, etc.)
   - Usefulness for the task
   - Type of distribution (Gaussian, uniform, logarithmic, etc.)
   - Do a histogram for each attribute.
4. For supervised learning tasks, identify the target attribute(s); i.e. the label(s).
5. Discover and visualize the data by scatter plots for each numerical attribute.
6. Study the correlations between attributes.
   Make also a scatter matrix plot.
7. N/A. Study how you would solve the problem manually.
8. Experiment with attribute combinations.
9. Identify a new promising attribute you may want to apply, if any.
10. N/A. Identify extra data that would be useful (go back to "Get the Data").
11. Create a test set, put it aside, and never look at it (no data snooping!).
    A problem here is that the data set is small. How much do you want in the test set…
    Remember *stratified*….
    Note. This would normally be done earlier, but the data size is so small.
12. *Document your work.*

Step 4. Prepare the data

You are now ready to clean and prepare the data.

Notes:

- Work on copies of the data (keep the original dataset intact).
- Write functions for all data transformations you apply, for five reasons:
  - So you can easily prepare the data the next time you get a fresh dataset
  - So you can apply these transformations in future projects
  - To clean and prepare the test set
  - To clean and prepare new data instances once your solution is live
  - To make it easy to treat your preparation choices as hyperparameters

1. Data cleaning:
   - N/A. Fix or remove outliers (optional).
   - Fill in missing values (e.g., with zero, not mean or median…)
     N/A or drop their rows (or columns).
2. Feature selection (optional):
   - Drop the attributes that provide no useful information for the task.
3. N/A. Feature engineering, where appropriate:
   - N/A. Discretize continuous features.
   - N/A. Decompose features (e.g., categorical, date/time, etc.).
   - N/A. Add promising transformations of features (e.g., log(x), sqrt(x), x^2, etc.).
   - N/A. Aggregate features into promising new features. Don't !
4. N/A. Handle text and categorical attributes using *"import OneHotEncoder"*.
5. Feature scaling:
   - Standardize or normalize features, if necessary.
6. *Document your work.*

Step 5. Select and train a model

Notes:

- N/A. If the data is huge, you may want to sample smaller training sets so you can train many different models in a reasonable time (be aware that this penalizes complex models such as large neural nets or Random Forests).
- Once again, try to automate these steps as much as possible.

1. Select a linear model using standard parameters.
2. Measure and compare the performance.
    - For each model (you only have the linear one), compute the mean and the root mean square of the performance measure on a manually selected subset (5-10 data) of the training data.
3. Analyze the most significant variables for each algorithm.
4. Analyze the types of errors the model makes.
    - What data would a human have used to avoid these errors?
5. Perform a quick round of feature selection and engineering.
6. N/A. Consider another quick-and-dirty model from different categories (e.g., naive Bayes, SVM, Random Forest, neural net, etc.).
7. N/A. Perform one or two more quick iterations of the five previous steps.
8. N/A. Shortlist the top one to two most promising models, preferring models that make different types of errors.
9. Later after *Step 6 "Fine tune"* and if you have time, consider how to measure and compare the performance in a more advanced way:
    - For each model (you only have the linear one), use *N*-fold cross-validation (*scoring="neg_mean_squared_error", cv=5*) and compute the mean and standard deviation of the performance measure on the *N* folds.
10. *Document your work.*

Step 6: Fine tune and test the model

Notes:

- You will want to use as much data as possible for this step, especially as you move toward the end of fine-tuning.
- As always, automate what you can.

1. N/A. Fine-tune the hyperparameters using cross-validation:
    - Treat your data transformation choices as hyperparameters, especially when you are not sure about them (e.g., if you're not sure whether to replace missing values with zeros or with the median value, or to just drop the rows).
    - Unless there are very few hyperparameter values to explore, prefer random search over grid search. If training is very long, you may prefer a Bayesian optimization approach (e.g., using Gaussian process priors, as described by Jasper Snoek et al.).
2. N/A. Try Ensemble methods. Combining your best models will often produce better performance than running them individually.
3. Once you are confident about your final model, measure its performance on the test set to estimate the generalization error. This is important!
4. *Document your work.*

Extra for the fast ones

1. Later after Step 6 "Fine tune" and if you have time, consider "Feature Scaling" and also how to measure and compare the performance in a more advanced way (mentioned in Step 5).
2. Later after Step 6 "Fine tune" and if you have time, consider other models (SVM, Random Forest) and compare the performance (mentioned in Step 5 point 6).
3. Later after Step 6 "Fine tune" and if you have time, consider "OneHotEncoder" (mentioned in step 4 point 4).

.

Step 7. Report and presentation

1. Document what you have done.
2. Create a nice and short introduction
    - Make sure you highlight the big picture first.
3. Explain why your solution achieves the business objective.
4. Do not forget to present interesting points you noticed along the way.
    - Describe what worked and what did not.
    - List your assumptions and your system's limitations.

5. N/A Ensure your key findings are communicated through beautiful visualizations or easy-to-remember statements (e.g., “the median income is the number-one predictor of housing prices”).
6. Upload your program together with the report in a .zip file in Wiseflow.