***Project 4: Web Scraping Job Postings***

By Krisdan

For this task I investigated 1079 job descriptions from seek.com in hope to achieve the following:

* Build a prediction model to target salary
* Build a prediction model to target if salary will be above, or below $90k
* Build a prediction model to target specific words in the job title

**Building a prediction model to target salary**

**Challenges**

* Many job listings did not include salary

**Solutions**

* Use Regex to extract numerical characters
  + Arrange by:
    - Daily
    - Monthly
    - Yearly
  + Pro-Rata to Yearly
* Fill all unknowns with the average salary of $90,000

**Results**

* Fitted model LinearRegression() with a negative R2 score of - 475 quintillion
* Might need to or source even more job-postings to offset the scarcity of salary information

**Build a prediction model to target if salary will be above, or below $90k**

**Challenges**

* Kurtosis in distribution due the scare salary data, when creating into three target-classifications:
  + Above $90k
  + On $90k
  + Below $90k
* High baseline accuracy of 90.09%

**Solutions**

* Calculate the baseline accuracy
* Use a range of predictive models
  + LogisticRegression(max\_iter=10000)
  + svm.SVC(kernel='linear')
  + KNeighborsClassifier
  + BaggingClassifier
* Find the most common words in each of the three target-categories
* Once a model didn’t exceed baseline accuracy, investigation ceases and another model is sought

**Results**

* None of the four models fitted could exceed baseline accuracy so no further investigation was made
* Able to rank most common words per target category and manually isolate the words specific to the target-category

|  |  |  |
| --- | --- | --- |
| **Most common words per Classification as Over/Over/At Average** | | |
| Above $90k | At $90k (Or Null Value) | Below $90k |
|  |  |  |
| science | reporting | requirements |
| learning | people | support |
| statistical | support | management |
| machine | analysis | reporting |
| analytics | insights | intelligence |

**Build a prediction model to target specific words in the job title**

**Challenges**

* Job Titles were varied

**Solutions**

* Built a function to categorise depending on the mention of specific words
  + 'science'
  + 'scientist'
  + 'analyst'
  + 'analysts'
  + 'analytical'
  + 'analytics
  + 'business'
  + 'intelligence'

**Results**

* Baseline Accuracy was only 51.81% (Much to my relief)
* All three models exceeded the Baseline Accuracy
* LogisticRegression() was the highest at 86.20%
* I was again able to rank most common words per target-category and manually isolate the words specific to the target-category

|  |  |  |
| --- | --- | --- |
| ***Most Common Words per Job Title*** | | |
| ***data scientist*** | ***data analyst*** | ***business intelligence*** |
| Learning  Machine  Statistical | Reporting  People  Support | Requirements  Solutions  Development |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model overview** | | | | | | |
| ***Section*** | ***Model*** | ***Target-Variable Type*** | ***Target-Variable comments*** | ***Score*** | ***Why This Model*** | ***My Comments*** |
| 1 | LinearRegression() | Continuous | Use Regex to decipher salaries  Fill nulls with $90kFill unknowns with $90k | R2 = - 475 quintillion | Starting-Point Model | Negative r2\_score means my prediction tends to be less accurate than the average value of the data ($128,405.99) |
| 1 | baseline\_accuracy | Discrete | Split Salaries into three groups: Above, On, Below $90k | 90.09% | baseline\_accuracy |  |
| 1 | LogisticRegression(max\_iter=10000) | (Same as above) | (Same as above) | 89.35% | Uses train\_test\_split | Hard to beat such a baseline accuracy |
| 1 | svm.SVC(kernel='linear') | (Same as above) | (Same as above) | 88.63% |  | Hard to beat such a baseline accuracy |
| 1 | KNeighborsClassifier() | (Same as above) | (Same as above) | 88.04% | Used for bagging | KNN performed surprisingly well given the number of features: 14,136 |
| 1 | BaggingClassifier(base\_estimator = KNeighborsClassifier(), max\_samples=0.5, max\_features=0.5) | (Same as above) | (Same as above) | 88.88% | Task Requirement |  |
| 2 | baseline\_accuracy | Discrete | Three job titles = ['data scientist', 'data analyst', 'business intelligence'] | 51.81% | baseline\_accuracy |  |
| 2 | LogisticRegression() | (Same as above) | (Same as above) | 86.20% | Uses train\_test\_split | Simplicity seems best when it comes to the models all used for Question 2 |
| 2 | KNeighborsClassifier() | (Same as above) | (Same as above) | 66.26% | Used for bagging | KNN performed surprisingly well given the number of features: 14,136 |
| 2 | BaggingClassifier(base\_estimator = KNeighborsClassifier(), max\_samples=0.5, max\_features=0.5) | (Same as above) | (Same as above) | 68.49% | Task Requirement |  |