Home Work 4

Part 1 Entropy

1. The dataset shown below represents bank customers with 3 features and the label corresponding to each customer identifies whether they've defaulted or not.

HasJob	HasFamily	IsAbove30years	Defaulter
1	1	1	0
1	1	1	0
1	0	1	0
0	1	0	0
0	0	1	1
0	1	0	1
1	0	1	1
1	0	1	1

Informations gain for:
Has Job= 0.04879494069539847
Has Family= 0.18872187554086717
Above 30 Years= 0.0
Since, IG for Has Family is the Greatest, it should be used for the First split.

<u>Explanation:</u>

Total number of entities= 8
Total number of defaulters= 0: 4
Total number of defaulters= 1: 4
Entropy= -((4/8)*math.log2(4/8))-((4/8)*math.log2(4/8))

#FOR HAS JOB:
# P(Has Job= 0) = 3/8
#For P(has Job=0)= 0:
#P(Defaulter_Has Job= 0)=1/3
#P(Defaulter_Has Job= 1)=2/3
#CONDITIONAL_ENTROPY=
(3/8)*(-(1/3)*math.log2(1/3)-(2/3)*math.log2(2/3))

# P(Has Job=1) = 5/8
#For P(has Job=1)= 1:
#P(Defaulter_Has Job= 0)=3/5
#P(Defaulter_Has Job= 1)=2/5
#CONDITIONAL_ENTROPY=
(5/8)*(-(3/5)*math.log2(3/5)-(2/5)*math.log2(2/5))
Overall:
Conditional Entropy:
(3/8)*(-(1/3)*math.log2(1/3)-(2/3)*math.log2(2/3))+ (5/8)*
(-(3/5)*math.log2(3/5)-(2/5)*math.log2(2/5))
= 0.9512050593046015
Has_Job_IG = Entropy- Conditional_Entropy(Has_Job)
Has_Job_IG = 0.04879494069539847

#FOR Has Family:
# P(Has Family= 0) = 4/8
#For P(has family=0)= 0:
#P(Defaulter_Has family= 0)=3/4
#P(Defaulter_Has family= 1)=1/4
#CONDITIONAL_ENTROPY=
(4/8)*(-(3/4)*math.log2(3/4)-(1/4)*math.log2(1/4))

# P(Has family=1) = 4/8
#For P(has family=1)= 1:
#P(Defaulter_Has family= 0)=3/4
#P(Defaulter_Has family= 1)=1/4
#CONDITIONAL_ENTROPY=
(4/8)*(-(1/4)*math.log2(1/4)-(3/4)*math.log2(3/4))
Overall:
Conditional Entropy: (4/8)*(-(3/4)*math.log2(3/4)-(1/4)*math.log2(1/4))+
(4/8)*(-(1/4)*math.log2(1/4)-(3/4)*math.log2(3/4))
=0.8112781244591328
Has_Family_IG = Entropy-Conditional_Entropy(Has_Family)
Has_Family_IG = 0.18872187554086717
#FOR 30years:
# P(30years= 0) = 2/8
#For P(30years=0)= 0:
#P(Defaulter_30years= 0)=1/2
#P(Defaulter_30years= 1)=1/2
#CONDITIONAL_ENTROPY=
(2/8)*(-(1/2)*math.log2(1/2)-(1/2)*math.log2(1/2))

# P(30years=1) = 6/8
#For P(30years=1)= 1:
#P(Defaulter_30years= 0)=3/6
#P(Defaulter_30years= 1)=3/6
#CONDITIONAL_ENTROPY=
(6/8)*(-(3/6)*math.log2(3/6)-(3/6)*math.log2(3/6))

Overall:
Conditional Entropy:
(2/8)*(-(1/2)*math.log2(1/2)-(1/2)*math.log2(1/2))+ (6/8)*
(-(3/6)*math.log2(3/6)-(3/6)*math.log2(3/6))
= 1.0
IG__30years = Entropy-Conditional_Entropy(30years)
IG__30years = 0.0

2. Given a signal of three symbols S=(A, B, C) and P(A)=0.7, P(B)=0.2, P(C)=0.1, What is the entropy of S? What does it mean according to the Source coding Theorem?

Entropy= 7/10 * math.log2(10/7) + 3/10 * math.log2(10/3) + 1/10 * math.log2(10) Entropy is: 1.213483708719429.
This means according to the source coding theorem that theoretically the smallest codworth length for the signal (S) is 1.21 bits per symbol.

Part 2 NLP

1. What is the difference between a Bag Of words Model in NLP and a Word2vec Model, discuss advantages of one over the other? (in 200 words)

Ans : When using the Bag of Words model, for a given document, only unique words taken that create an unordered list of the words in that document. These words form the bag of words that represent the original document. This bag is used to create a numeric representation of the words in the document by counting how many times a word appears in that document.

Cons: Semantic meaning of the words is lost The ordering of words is lost

Pros: The output is often still useful/accurate.

The Word2Vec model uses distributional representations of words: words with similar meanings cluster together in a way relationships between words can be represented mathematically.

Cons: Sense of the word is not captured such that "cell" could mean biological, prison etc.

Pros: Labels are not needed - you can just feed a corpus and it will output vectors

2. What is a word vector? What is a word Embedding? On what factors does the word embedding of a word depend (explain it from a NLP perspective)? (in 100 words)

Ans: A word vector is a representation of a word in a n-dimensional space such that n sufficiently encodes the semanatics of the document that the word is in. A word embedding is a parameterised function that maps the words to the vectors. It quantifies the semantic similarities between linguistic items based on their distributional properties in large documents. So, the frequency of the words and the ways in which the words interact with each other linguistically in the sentences would both influence the word embedding.

3. What is a corpus in NLP? How is the vocabulary of a model different from the corpus? (in 50 words)

Ans: A corpus is the large set of documents, usually organized in some methodical way, which is used as the input for the processing. A vocabulary shows the frequency counts of unique words in the corpus: it is usually unordered, hence semantic meaning on first inspection is removed.

4. Train a word2vec model on the corpus consisting of the text in the novel Pride and Prejudice.

Ans:

1. Regex was used to remove all special characters (only alphabetical remaining)

2. Strings were converted to lower case

3. Strings were tokenized using PorterStemmer (if lemmatize = True) and/or WordNetLemmatizer (if stem = True)

4. English Stopwords were removed For the model below, only lemmatize = True, i.e. only PorterStemmer was used. Processing: Used GenSim's word2vec package with the following hyperparameter settings: size = 100, window = 5, min_count = 5, workers = 3, iter = 100.

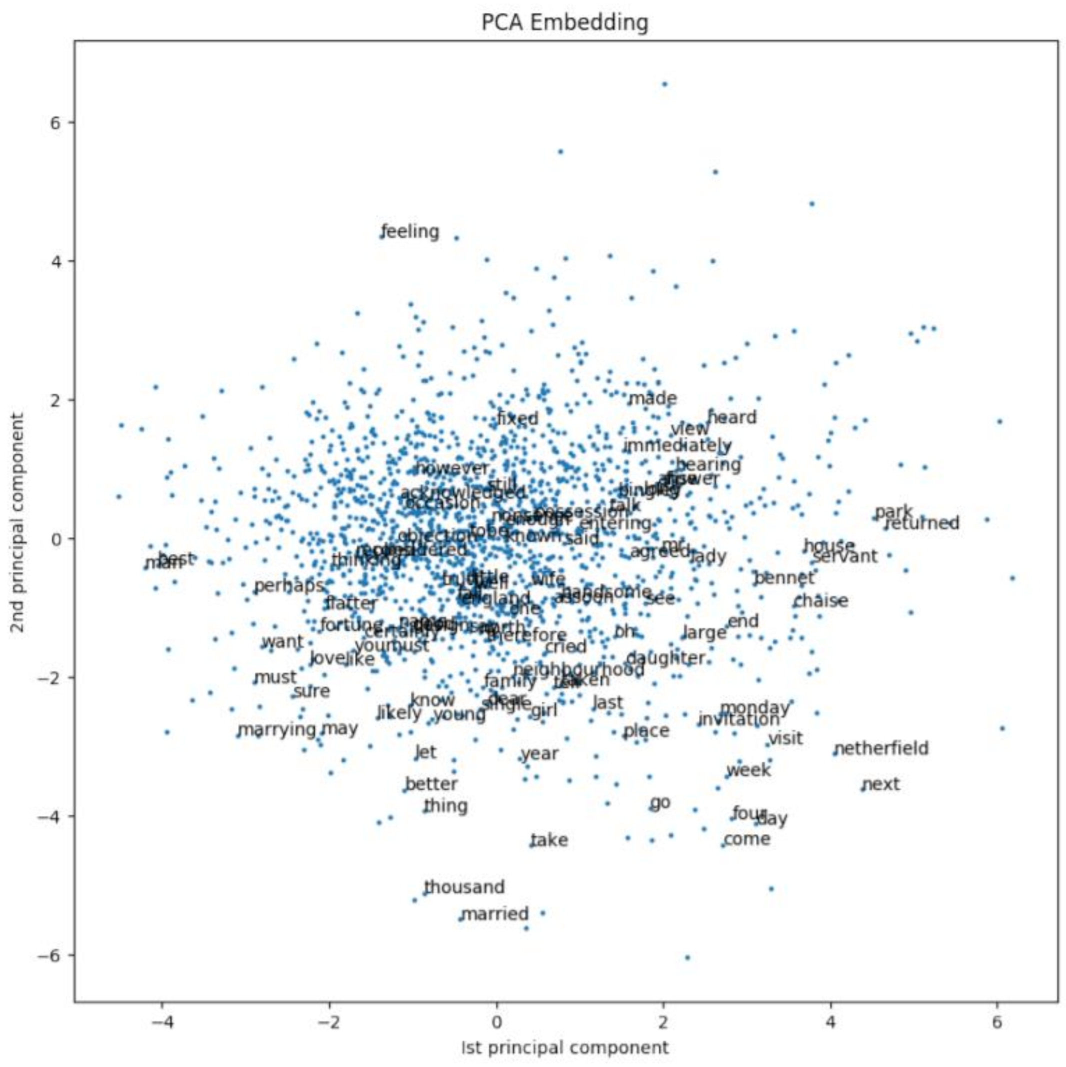
There were 3 worker threads to train the model over 100 training iterations over the corpus, which used a maximum distance between the current and predicted word within a sentence of 5. This produced a model with dimensionality of 100, ignoring all the words with total frequency less than 5. Results: The embedding model had a shape of (1745, 100). That is, it transformed 1745 words (the vocabulary) in the sentences to 100- dimensional vectors.

The Vocabulary:

['truth', 'acknowledged', 'single', 'man', 'good', 'fortune', 'must', 'want', 'wife', 'however', 'little', 'known', 'feeling', 'view', 'may', 'entering', 'neighbourhood', 'well', 'fixed', 'family', 'considered', 'one', 'daughter', 'dear', 'mr', 'bennet', 'said', 'lady', 'day', 'heard', 'park', 'let', 'last', 'replied', 'returned', 'long', 'made', 'answer', 'know', 'taken', 'cried', 'tell', 'objection', 'hearing', 'invitation', 'enough', 'say', 'netherfield', 'young', 'large', 'north', 'england', 'monday', 'chaise', 'four', 'see', 'place', 'agreed', 'immediately', 'take', 'possession', 'servant', 'tobe', 'house', 'end', 'next', 'week', 'name', 'bingley', 'married', 'oh', 'sure', 'thousand', 'year', 'fine', 'thing', 'girl', 'youmust', 'thinking', 'marrying', 'design', 'nonsense', 'talk', 'likely', 'fall', 'love', 'therefore', 'visit', 'assoon', 'come', 'occasion', 'go', 'perhaps', 'still', 'better', 'handsome', 'like', 'best', 'flatter', 'certainly', 'share', 'beauty', 'buti', 'pretend', 'anything', 'extraordinary', 'woman', 'ought', 'give', 'case', 'often', 'much', 'think', 'indeed', 'assure', 'consider', 'establishment', 'wouldbe', 'sir', 'william', 'lucas', 'determined', 'merely', 'account', 'general', 'impossible', 'u', 'dare', 'send', 'line', 'consent', 'chooses', 'throw', 'word', 'lizzy', 'desire', 'half', 'jane', 'humoured', 'lydia', 'always', 'giving', 'preference', 'none', 'recommend', 'silly', 'ignorant', 'something', 'sister', 'child', 'way', 'delight', 'compassion', 'poor', 'mistake', 'high', 'respect', 'old', 'friend', 'mention', 'twenty', 'least', 'ah', 'suffer', 'hope', 'get', 'live', 'many', 'men', 'use', 'since', 'depend', 'upon', 'odd', 'mixture', 'quick', 'part', 'humour', 'reserve', 'three', 'hadbeen', 'insufficient', 'make', 'understand', 'character', 'le', 'difficult', 'mean', 'understanding', 'information', 'temper', 'fancied', 'business', 'life', 'visiting', 'news', 'among', 'earliest', 'waited', 'hehad', 'intended', 'though', 'till', 'evening', 'knowledge', 'observing', 'second', 'employed',

```
'addressed', 'forget', 'mamma', 'elizabeth', 'shall', 'meet', 'promised', 'introduce', 'believe', 'two', 'selfish',
'glad', 'find', 'reply', 'unable', 'began', 'keep', 'kitty', 'heaven', 'sake', 'tear', 'piece', 'discretion', 'father',
'ill', 'amusement', 'ball', 'morrow', 'fortnight', 'aye', 'mother', 'advantage', 'acquainted', 'teasing',
'honour', 'acquaintance', 'cannot', 'really', 'else', 'stand', 'chance', 'asshe', 'act', 'kindness', 'office',
'stared', 'meaning', 'exclamation', 'form', 'introduction', 'laid', 'quite', 'agree', 'mary', 'deep', 'reflection',
'book', 'wished', 'sensible', 'knew', 'idea', 'continued', 'return', 'sick', 'sorry', 'hear', 'morning', 'would',
'unlucky', 'actually', 'paid', 'escape', 'astonishment', 'rest', 'first', 'declare', 'expected', 'loved', 'pleased',
'agood', 'gone', 'never', 'choose', 'spoke', 'left', 'room', 'rapture', 'excellent', 'door', 'ever', 'either',
'matter', 'time', 'making', 'new', 'every', 'youngest', 'dance', 'afraid', 'spent', 'soon', 'ask', 'assistance',
'could', 'subject', 'draw', 'satisfactory', 'various', 'question', 'supposition', 'obliged', 'accept', 'hand',
'intelligence', 'neighbour', 'report', 'highly', 'favourable', 'whole', 'meant', 'party', 'nothing', 'delightful',
'fond', 'certain', 'step', 'towards', 'heart', 'entertained', 'happily', 'settled', 'husband', 'others', 'equally',
'wish', 'sat', 'minute', 'library', 'sight', 'whose', 'saw', 'somewhat', 'coat', 'horse', 'dinner', 'afterwards',
'course', 'credit', 'arrived', 'bingleywas', 'town', 'following', 'consequently', 'etc', 'imagine', 'arrival',
'hertfordshire', 'fear', 'thathe', 'might', 'another', 'london', 'followed', 'bring', 'gentleman', 'assembly',
'grieved', 'instead', 'brought', 'five', 'cousin', 'entered', 'eldest', 'looking', 'gentlemanlike', 'easy',
'manner', 'air', 'decided', 'fashion', 'brother', 'law', 'hurst', 'darcy', 'drew', 'tall', 'person', 'feature', 'noble',
'andthe', 'within', 'entrance', 'ten', 'figure', 'declared', 'hewas', 'looked', 'gave', 'turned', 'discovered',
'company', 'estate', 'derbyshire', 'disagreeable', 'countenance', 'lively', 'danced', 'angry', 'closed', 'early',
'talked', 'amiable', 'quality', 'speak', 'miss', 'declined', 'speaking', 'occasionally', 'world', 'everybody',
'hoped', 'amongst', 'violent', 'dislike', 'ofhis', 'behaviour', 'particular', 'resentment', 'near', 'conversation',
'came', 'join', 'hate', 'unless', 'partner', 'engaged', 'punishment', 'met', 'pleasant', 'several', 'pretty',
'dancing', 'beautiful', 'creature', 'sitting', 'behind', 'agreeable', 'turning', 'round', 'moment', 'eye',
'tolerable', 'present', 'consequence', 'enjoy', 'advice', 'walked', 'told', 'story', 'great', 'spirit', 'disposition',
'delighted', 'altogether', 'passed', 'seen', 'admired', 'twice', 'gratified', 'felt', 'mentioned', 'catherine',
'without', 'theyhad', 'yet', 'learnt', 'care', 'longbourn', 'village', 'lived', 'principal', 'found', 'deal',
'curiosity', 'event', 'expectation', 'rather', 'onthe', 'stranger', 'disappointed', 'thought', 'asked', 'vexed',
'admire', 'seemed', 'struck', 'inquired', 'got', 'introduced', 'third', 'king', 'maria', 'charming', 'elegant',
'dress', 'gown', 'interrupted', 'protested', 'seek', 'related', 'bitterness', 'added', 'lose', 'fancy', 'worth',
'pleasing', 'given', 'set', 'alone', 'former', 'inher', 'praise', 'expressed', 'happy', 'perfect', 'breeding', 'also',
'possibly', 'complete', 'flattered', 'asking', 'expect', 'compliment', 'difference', 'surprise', 'natural',
'thanks', 'leave', 'liked', 'people', 'fault', 'anybody', 'anyone', 'wonder', 'sense', 'blind', 'folly', 'common',
'bad', 'equal', 'mistaken', 'listened', 'silence', 'convinced', 'please', 'observation', 'judgement', 'attention',
'shewas', 'disposed', 'fact', 'chose', 'proud', 'thefirst', 'private', 'habit', 'rank', 'respectable',
'circumstance', 'memory', 'property', 'nearly', 'pound', 'purchase', 'likewise', 'choice', 'provided', 'liberty',
'whether', 'spend', 'anxious', 'established', 'unwilling', 'table', 'home', 'tempted', 'look', 'hour', 'situation',
'satisfied', 'took', 'steady', 'friendship', 'spite', 'offer', 'greater', 'appeared', 'strength', 'regard', 'highest',
'opinion', 'superior', 'clever', 'reserved', 'greatly', 'wherever', 'offense', 'meryton', 'kind', 'contrary',
'smallest', 'interest', 'pleasure', 'smiled', 'allowed', 'pronounced', 'sweet', 'object', 'short', 'walk',
'particularly', 'intimate', 'formerly', 'address', 'strongly', 'small', 'quitting', 'removed', 'mile', 'period',
'lodge', 'importance', 'civil', 'allthe', 'friendly', 'obliging', 'presentation', 'st', 'james', 'absolutely',
'necessary', 'charlotte', 'command', 'yes', 'suppose', 'seem', 'hardly', 'beyond', 'doubt', 'point', 'purpose',
'eliza', 'listening', 'beg', 'put', 'head', 'misfortune', 'night', 'close', 'opening', 'lip', 'help', 'guess', 'itwas',
'pride', 'carriage', 'tothe', 'mind', 'talking', 'promise', 'excuse', 'everything', 'favour', 'express', 'right',
'true', 'easily', 'forgive', 'mortified', 'mine', 'observed', 'read', 'self', 'real', 'imaginary', 'vanity', 'used',
'maybe', 'vain', 'rich', 'drink', 'away', 'boy', 'argument', 'due', 'grew', 'younger', 'received', 'even', 'value',
'probability', 'influence', 'admiration', 'generally', 'evident', 'whenever', 'composure', 'cheerfulness',
'guard', 'impertinent', 'able', 'public', 'sometimes', 'disadvantage', 'guarded', 'affection', 'opportunity',
'consolation', 'inthe', 'gratitude', 'almost', 'safe', 'begin', 'slight', 'show', 'feel', 'undoubtedly', 'nature',
'allow', 'remember', 'partial', 'endeavour', 'tolerably', 'together', 'secure', 'leisure', 'plan', 'degree',
'dined', 'unfolded', 'success', 'happiness', 'marriage', 'entirely', 'felicity', 'continue', 'vexation', 'defect',
'pas', 'laugh', 'sound', 'far', 'clear', 'face', 'expression', 'discovery', 'succeeded', 'mortifying', 'forced',
'light', 'caught', 'perfectly', 'attended', 'notice', 'colonel', 'forster', 'approaching', 'intention', 'sucha',
'energy', 'severe', 'turn', 'open', 'instrument', 'follows', 'strange', 'wanting', 'meto', 'play', 'sing', 'sit',
'saying', 'song', 'performance', 'entreaty', 'thatshe', 'eagerly', 'hersister', 'plain', 'hard', 'accomplishment',
'neither', 'taste', 'shehad', 'reached', 'playing', 'request', 'officer', 'joined', 'stood', 'silent', 'indignation',
'mode', 'perceive', 'thus', 'thereis', 'society', 'pause', 'seeing', 'proper', 'pay', 'avoid', 'companion',
'instant', 'moving', 'called', 'toher', 'desirable', 'refuse', 'taking', 'surprised', 'receive', 'back', 'entreat',
'moved', 'order', 'grave', 'propriety', 'ofher', 'attempt', 'persuasion', 'oblige', 'politeness', 'smiling',
'considering', 'inducement', 'conjecture', 'totally', 'wrong', 'bestow', 'desired', 'hewould', 'repeated',
'favourite', 'pray', 'exactly', 'matrimony', 'wishing', 'joy', 'nay', 'serious', 'pemberley', 'indifference', 'wit',
'entailed', 'relation', 'phillips', 'distance', 'usually', 'duty', 'aunt', 'shop', 'frequent', 'vacant', 'offered',
'amuse', 'country', 'learn', 'recent', 'militia', 'regiment', 'remain', 'winter', 'connection', 'length', 'opened',
'niece', 'unknown', 'suspected', 'going', 'thenext', 'astonished', 'sentiment', 'oftheir', 'age', 'six',
'standing', 'prevented', 'note', 'calling', 'haste', 'aloud', 'dine', 'louisa', 'danger', 'tete', 'thegentlemen',
'caroline', 'dining', 'seems', 'stay', 'scheme', 'coach', 'spare', 'wanted', 'answered', 'hermother', 'cheerful',
'uneasy', 'lucky', 'aware', 'breakfast', 'scarcely', 'dearest', 'getting', 'yesterday', 'returning', 'insist', 'jones',
'alarmed', 'comfort', 'pursuit', 'ofmr', 'resolution', 'hint', 'motive', 'reason', 'exertion', 'proportion',
'required', 'accepted', 'along', 'parted', 'impatient', 'finding', 'warmth', 'exercise', 'shown', 'parlour',
'appearance', 'weather', 'therewas', 'complexion', 'asto', 'coming', 'latter', 'inquiry', 'missbennet',
'inconvenience', 'expressing', 'besides', 'treated', 'showed', 'supposed', 'cold', 'bed', 'readily', 'symptom',
'increased', 'quit', 'clock', 'parting', 'clothes', 'past', 'summoned', 'solicitude', 'excessively', 'enjoyment',
'anxiety', 'believed', 'card', 'prefer', 'directly', 'impertinence', 'style', 'wild', 'picture', 'lost', 'whatever',
'sort', 'independence', 'whisper', 'affected', 'speech', 'low', 'uncle', 'inmeryton', 'laughed', 'heartily',
'itwould', 'expense', 'renewal', ...]
```

The PCA Decomposition looks as below:



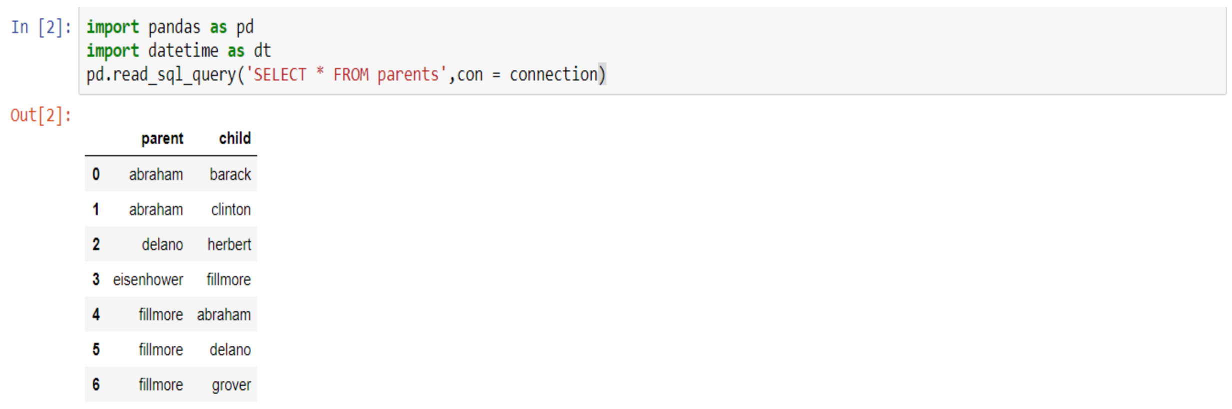
Using the doesnt_match method: For “boy girl man woman happy"; the model returns "happy".

Output: as expected. For ""father daughter son king""; the model returns "king". This is as expected. Using the similarity method: Between 'husband' and 'wife"; the cosine similarity is -0.026524968241996127. Between 'king' and 'man'; the cosine similarity= 0.2824083089901176

Part 3 SQL

1. Simple SELECTS (on the parents table)

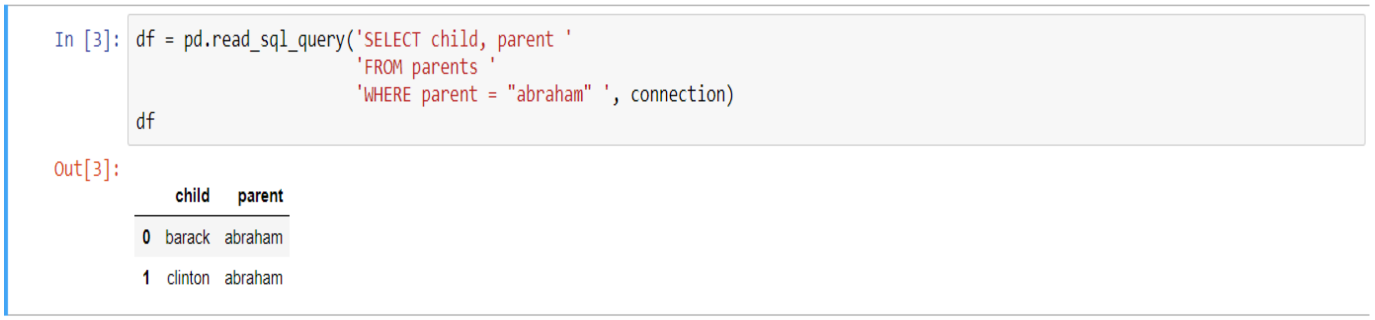
1. SELECT all records in the table.



```
In [2]: import pandas as pd
        import datetime as dt
        pd.read_sql_query('SELECT * FROM parents',con = connection)
Out[2]:
```

	parent	child
0	abraham	barack
1	abraham	clinton
2	delano	herbert
3	eisenhower	fillmore
4	fillmore	abraham
5	fillmore	delano
6	fillmore	grover

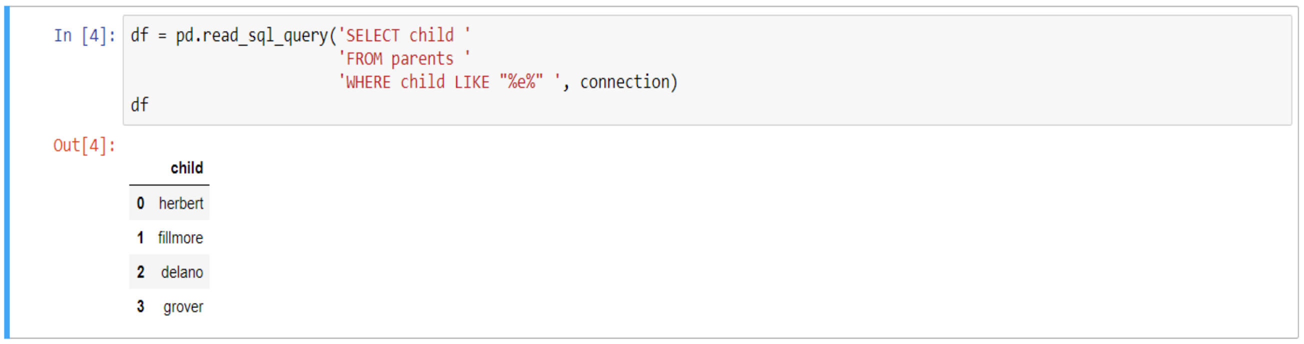
2. SELECT child and parent, where abraham is the parent.



```
In [3]: df = pd.read_sql_query('SELECT child, parent '
                               'FROM parents '
                               'WHERE parent = "abraham" ', connection)
        df
Out[3]:
```

	child	parent
0	barack	abraham
1	clinton	abraham

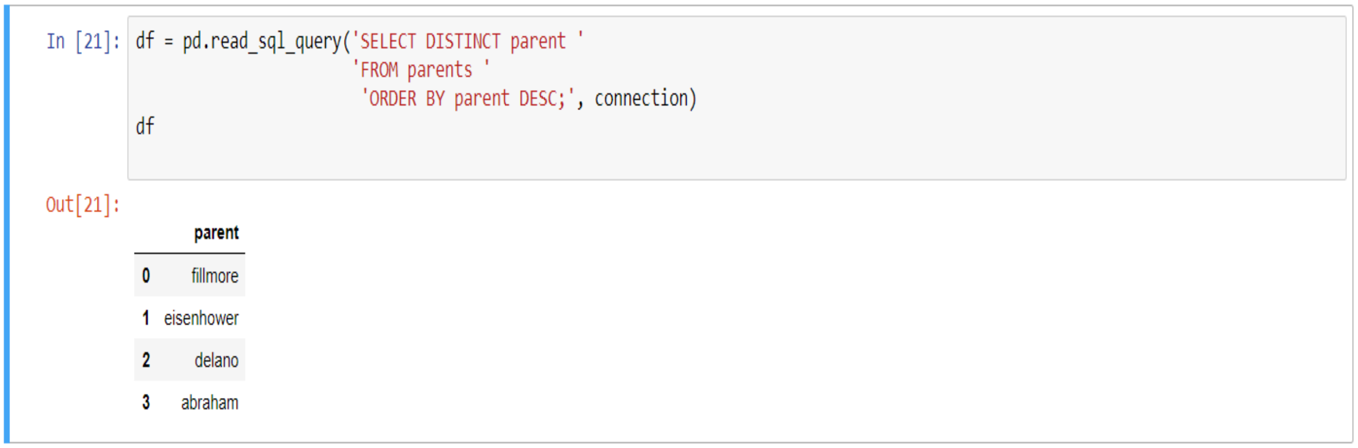
3. SELECT all children that have an 'e' in their name (hint: use LIKE and '%e%').



```
In [4]: df = pd.read_sql_query('SELECT child '
                               'FROM parents '
                               'WHERE child LIKE "%e%" ', connection)
        df
Out[4]:
```

	child
0	herbert
1	fillmore
2	delano
3	grover

4. SELECT all unique parents (use SELECT DISTINCT) and order them by name, descending order (i.e. fillmore first)

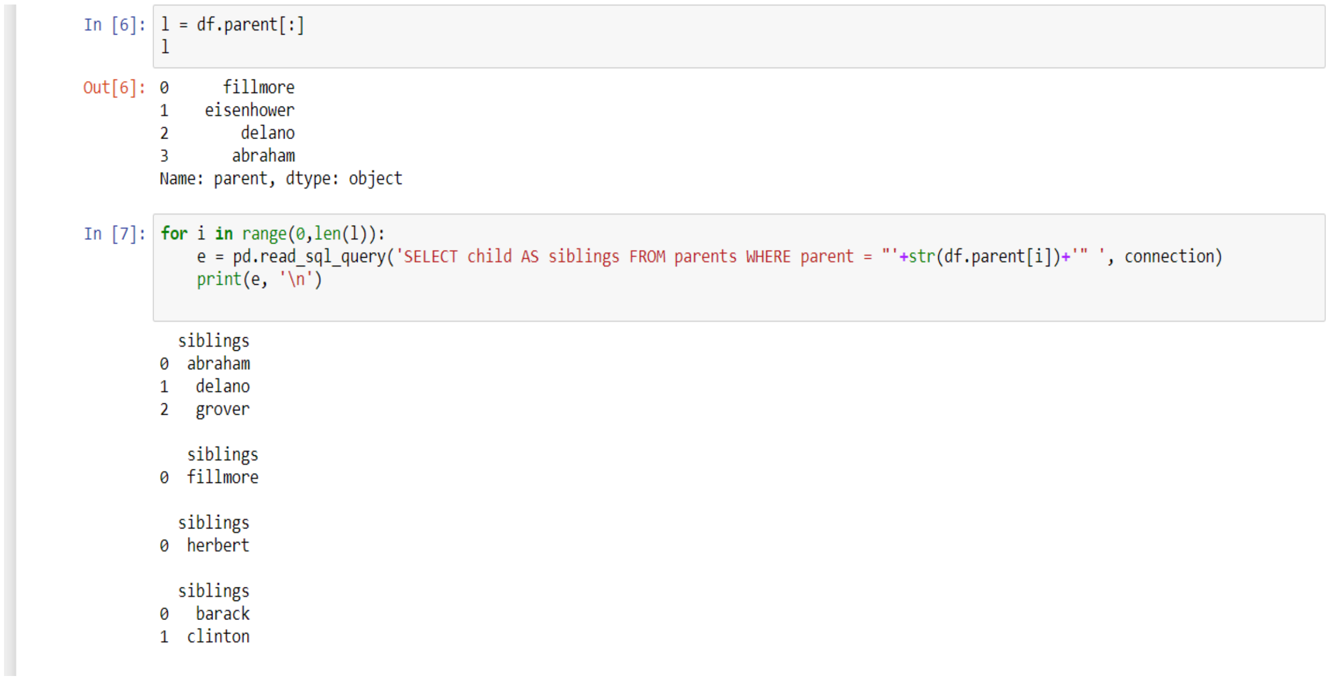


```
In [21]: df = pd.read_sql_query('SELECT DISTINCT parent '
                                'FROM parents '
                                 'ORDER BY parent DESC;', connection)
         df
```

Out[21]:

	parent
0	fillmore
1	eisenhower
2	delano
3	abraham

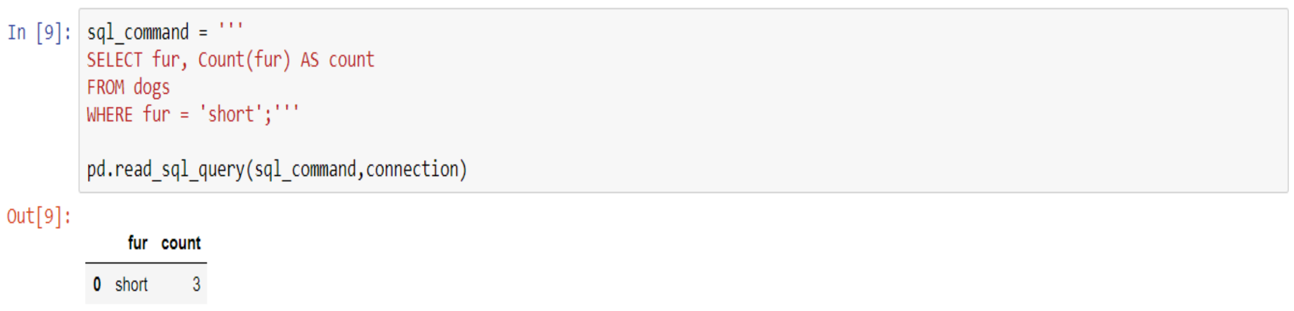
5. SELECT all dogs that are siblings (one-to-one relations). Only show a sibling pair once. To do this you need to select two times from the parents table.



```
In [6]: l = df.parent[:]
        l

Out[6]: 0      fillmore
        1    eisenhower
        2        delano
        3       abraham
        Name: parent, dtype: object

In [7]: for i in range(0,len(l)):
            e = pd.read_sql_query('SELECT child AS siblings FROM parents WHERE parent = "'+str(df.parent[i])+'" ', connection)
            print(e, '\n')

          siblings
        0  abraham
        1   delano
        2   grover

           siblings
        0  fillmore

          siblings
        0  herbert

          siblings
        0   barack
        1  clinton
```

2. JOINS

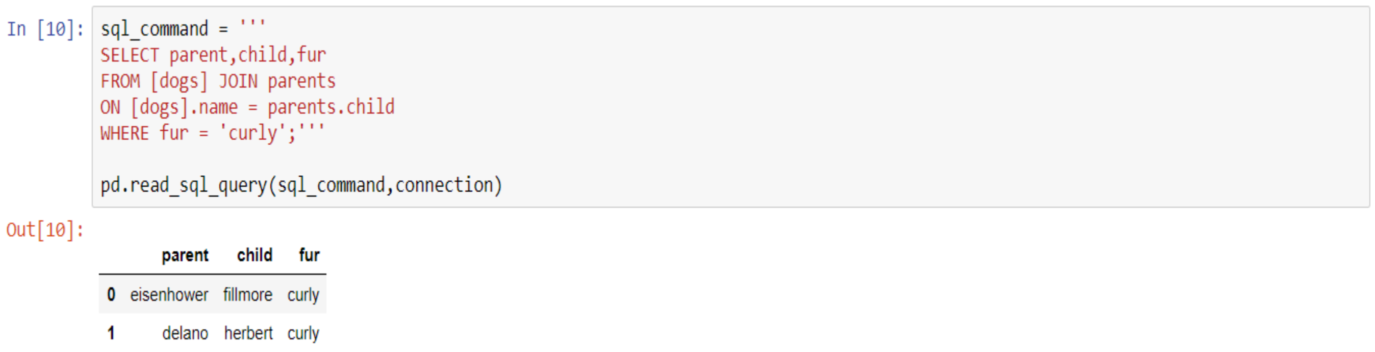
1. COUNT the number of short haired dogs



```
In [9]: sql_command = '''
        SELECT fur, Count(fur) AS count
        FROM dogs
        WHERE fur = 'short';'''

        pd.read_sql_query(sql_command,connection)
```

```
Out[9]:
     fur  count
0  short      3
```

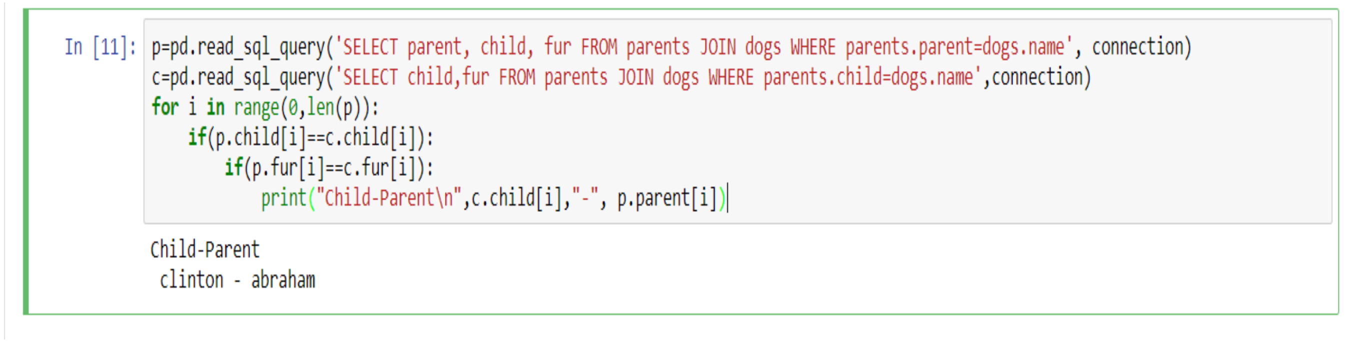
2. JOIN tables parents and dogs and SELECT the parents of curly dogs.



```
In [10]: sql_command = '''
         SELECT parent,child,fur
         FROM [dogs] JOIN parents
         ON [dogs].name = parents.child
         WHERE fur = 'curly';'''

         pd.read_sql_query(sql_command,connection)
```

```
Out[10]:
       parent     child    fur
0  eisenhower  fillmore  curly
1      delano    herbert  curly
```

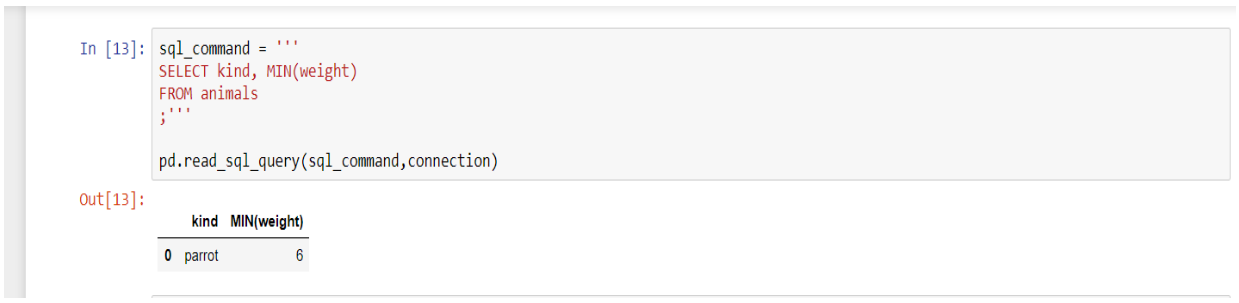
2. JOIN tables parents and dogs, and SELECT the parents and children that have the same fur type. Only show them once.



```
In [11]: p=pd.read_sql_query('SELECT parent, child, fur FROM parents JOIN dogs WHERE parents.parent=dogs.name', connection)
         c=pd.read_sql_query('SELECT child,fur FROM parents JOIN dogs WHERE parents.child=dogs.name',connection)
         for i in range(0,len(p)):
             if(p.child[i]==c.child[i]):
                 if(p.fur[i]==c.fur[i]):
                     print("Child-Parent\n",c.child[i],"-", p.parent[i])

         Child-Parent
          clinton - abraham
```

3. Aggregate functions, numerical logic and grouping

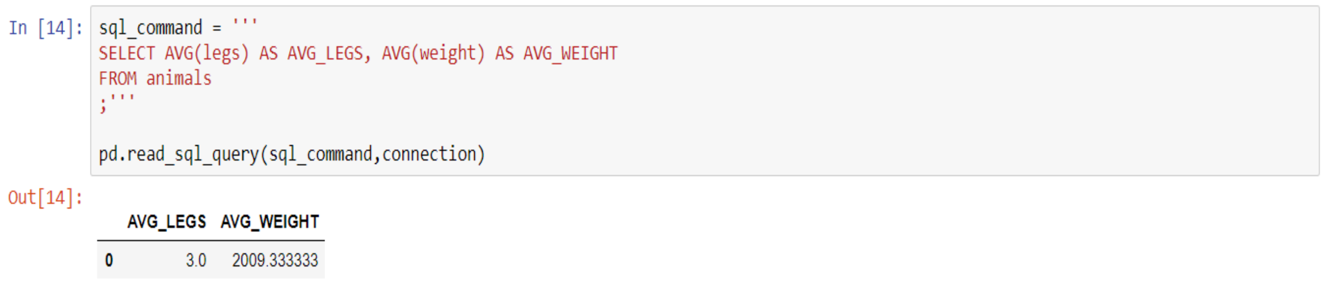
1. SELECT the animal with the minimum weight. Display kind and min_weight.



```
In [13]: sql_command = '''
         SELECT kind, MIN(weight)
         FROM animals
         ;'''

         pd.read_sql_query(sql_command,connection)
```

```
Out[13]:
     kind  MIN(weight)
0  parrot            6
```

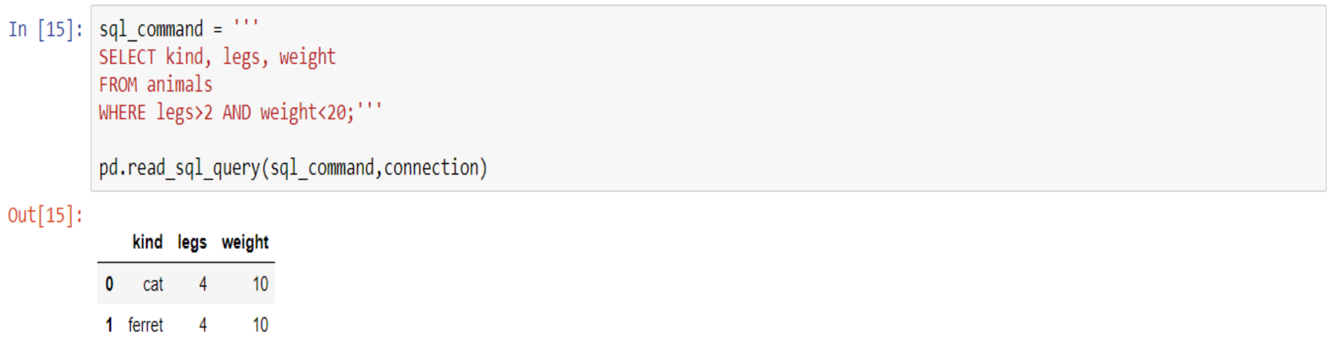
2. Use the aggregate function AVG to display a table with the average number of legs and the average weight.



```
In [14]: sql_command = '''
         SELECT AVG(legs) AS AVG_LEGS, AVG(weight) AS AVG_WEIGHT
         FROM animals
         ;'''

         pd.read_sql_query(sql_command,connection)
Out[14]:
```

	AVG_LEGS	AVG_WEIGHT
0	3.0	2009.333333

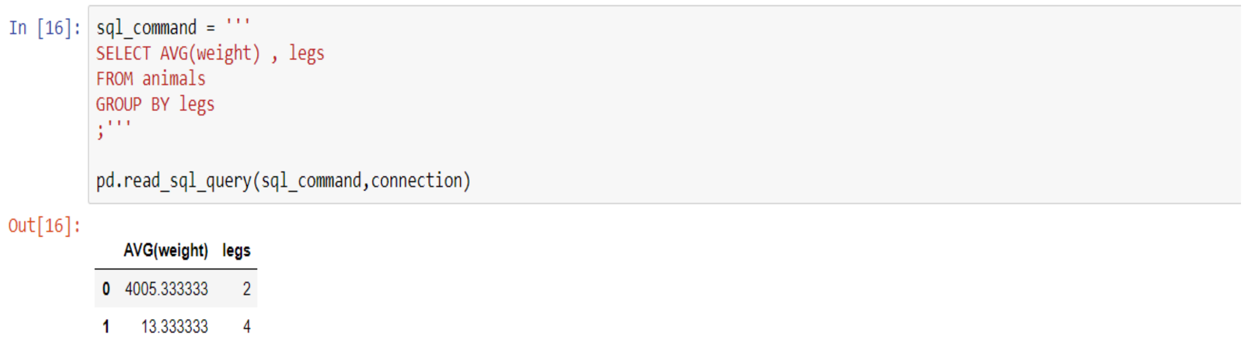
3. SELECT the animal kind(s) that have more than two legs, but weighs less than 20. Display kind, weight, legs.



```
In [15]: sql_command = '''
         SELECT kind, legs, weight
         FROM animals
         WHERE legs>2 AND weight<20;'''

         pd.read_sql_query(sql_command,connection)
Out[15]:
```

	kind	legs	weight
0	cat	4	10
1	ferret	4	10

4. SELECT the average weight for all the animals with 2 legs and the animals with 4 legs (by using GROUP BY).



```
In [16]: sql_command = '''
         SELECT AVG(weight) , legs
         FROM animals
         GROUP BY legs
         ;'''

         pd.read_sql_query(sql_command,connection)
Out[16]:
```

	AVG(weight)	legs
0	4005.333333	2
1	13.333333	4