index

May 6, 2022

1 Keras - Lab

1.1 Introduction

In this lab you'll once again build a neural network, but this time you will be using Keras to do a lot of the heavy lifting.

1.2 Objectives

You will be able to:

- Build a neural network using Keras
- Evaluate performance of a neural network using Keras

1.3 Required Packages

We'll start by importing all of the required packages and classes.

[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from sklearn import preprocessing
from keras.preprocessing.text import Tokenizer
from keras import models
from keras import layers
from keras import optimizers
```

1.4 Load the data

In this lab you will be classifying bank complaints available in the `'Bank_complaints.csv'` file.

[2]:
```
# Import data
df = pd.read_csv('Bank_complaints.csv')

# Inspect data
print(df.info())
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Data columns (total 2 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   Product                       60000 non-null  object
 1   Consumer complaint narrative  60000 non-null  object
dtypes: object(2)
memory usage: 937.6+ KB
None
```

[2]:
```
        Product                       Consumer complaint narrative
0  Student loan  In XX/XX/XXXX I filled out the Fedlaon applica…
1  Student loan  I am being contacted by a debt collector for p…
2  Student loan  I cosigned XXXX student loans at SallieMae for…
3  Student loan  Navient has sytematically and illegally failed…
4  Student loan  My wife became eligible for XXXX Loan Forgiven…
```

As mentioned earlier, your task is to categorize banking complaints into various predefined categories. Preview what these categories are and what percent of the complaints each accounts for.

[3]:
```python
# Your code here
df["Product"].value_counts(normalize = True)
```

[3]:
```
Student loan                   0.190067
Credit card                    0.159000
Consumer Loan                  0.157900
Mortgage                       0.138867
Bank account or service        0.138483
Credit reporting               0.114400
Checking or savings account    0.101283
Name: Product, dtype: float64
```

1.5 Preprocessing

Before we build our neural network, we need to do several preprocessing steps. First, we will create word vector counts (a bag of words type representation) of our complaints text. Next, we will change the category labels to integers. Finally, we will perform our usual train-test split before building and training our neural network using Keras. With that, let's start munging our data!

1.6 One-hot encoding of the complaints

Our first step again is to transform our textual data into a numerical representation. As we saw in some of our previous lessons on NLP, there are many ways to do this. Here, we'll use the `Tokenizer()` class from the `preprocessing.text` sub-module of the Keras package.

As with our previous work using NLTK, this will transform our text complaints into word vectors. (Note that the method of creating a vector is different from our previous work with NLTK; as you'll see, word order will be preserved as opposed to a bag of words representation). In the below code, we'll only keep the 2,000 most common words and use one-hot encoding.

[4]:
```
# As a quick preliminary, briefly review the docstring for keras.preprocessing.
↪text.Tokenizer
Tokenizer?
```

[5]:
```
#  This cell may take about thirty seconds to run

# Raw text complaints
complaints = df['Consumer complaint narrative']

# Initialize a tokenizer
tokenizer = Tokenizer(num_words=2000)

# Fit it to the complaints
tokenizer.fit_on_texts(complaints)

# Generate sequences
sequences = tokenizer.texts_to_sequences(complaints)
print('sequences type:', type(sequences))

# Similar to sequences, but returns a numpy array
one_hot_results= tokenizer.texts_to_matrix(complaints, mode='binary')
print('one_hot_results type:', type(one_hot_results))

# Useful if we wish to decode (more explanation below)
word_index = tokenizer.word_index

# Tokens are the number of unique words across the corpus
print('Found %s unique tokens.' % len(word_index))

# Our coded data
print('Dimensions of our coded results:', np.shape(one_hot_results))
```

```
sequences type: <class 'list'>
one_hot_results type: <class 'numpy.ndarray'>
Found 50110 unique tokens.
Dimensions of our coded results: (60000, 2000)
```

1.7 Decoding Word Vectors

As a note, you can also decode these vectorized representations of the reviews. The `word_index` variable, defined above, stores the mapping from the label number to the actual word. Somewhat tediously, we can turn this dictionary inside out and map it back to our word vectors, giving us roughly the original complaint back. (As you'll see, the text won't be identical as we limited ourselves to top 2000 words.)

1.8 Python Review / Mini Challenge

While a bit tangential to our main topic of interest, we need to reverse our current dictionary `word_index` which maps words from our corpus to integers. In decoding our `one_hot_results`, we will need to create a dictionary of these integers to the original words. Below, take the `word_index` dictionary object and change the orientation so that the values are keys and the keys values. In other words, you are transforming something of the form {A:1, B:2, C:3} to {1:A, 2:B, 3:C}.

[9]:
```python
# Your code here
reverse_index = {word_index[key]:key for key in word_index.keys()}
reverse_index

# G
# reverse_index = dict([(value, key) for (key, value) in word_index.items()])
```

[9]:
```
{1: 'xxxx',
 2: 'the',
 3: 'i',
 4: 'to',
 5: 'and',
 6: 'my',
 7: 'a',
 8: 'that',
 9: 'of',
 10: 'was',
 11: 'in',
 12: 'they',
 13: 'on',
 14: 'for',
 15: 'have',
 16: 'not',
 17: 'me',
 18: 'this',
 19: 'is',
 20: 'with',
 21: 'xx',
 22: 'it',
 23: 'account',
 24: '00',
 25: 'credit',
 26: 'from',
 27: 'had',
 28: 'as',
 29: 'be',
 30: 'loan',
 31: 'bank',
 32: 'would',
 33: 'payment',
 34: 'at',
 35: 'them',
 36: 'been',
 37: 'an',
 38: 'by',
 39: 'but',
 40: 'we',
 41: 'no',
 42: 'told',
 43: 'or',
 44: 'are',
 45: 'did',
 46: 'do',
 47: 'when',
 48: 'has',
 49: 'were',
 50: "n't",
 51: 'all',
 52: 'am',
 53: 'payments',
 54: 'which',
 55: 'card',
 56: 'time',
 57: 'so',
 58: 'their',
 59: 'called',
 60: 'received',
 61: 'after',
 62: 'because',
 63: 'pay',
 64: 'you',
 65: 'if',
 66: 'out',
 67: 'any',
 68: 'information',
 69: "''",
 70: 'can',
 71: 'get',
 72: 'there',
 73: 'back',
 74: "'s",
 75: 'will',
 76: 'never',
 77: 'could',
 78: 'up',
 79: 'she',
 80: 'then',
 81: 'money',
 82: 'made',
 83: 'said',
 84: 'about',
 85: 'now',
 86: 'over',
 87: 'due',
 88: 'interest',
 89: 'report',
 90: 'paid',
 91: 'he',
 92: 'what',
 93: 'call',
 94: 'amount',
 95: 'our',
 96: 'company',
 97: 'also',
 98: 'only',
 99: 'loans',
 100: 'one',
 101: 'check',
 102: 'balance',
 103: 'sent',
 104: 'mortgage',
 105: 'since',
 106: 'days',
 107: 'month',
 108: 'letter',
 109: 'make',
 110: 'being',
 111: 'late',
 112: 'asked',
 113: 'phone',
 114: 'years',
 115: 'still',
 116: 'again',
 117: 'even',
 118: 'should',
 119: 'off',
 120: 'fees',
 121: 'who',
 122: 'months',
 123: 'number',
 124: 'into',
 125: 'us',
 126: 'more',
 127: 'help',
 128: 'these',
 129: 'service',
 130: 'just',
 131: 'other',
 132: 'through',
 133: 'your',
 134: 'new',
 135: 'customer',
 136: 'day',
 137: 'chase',
 138: 'her',
 139: 'home',
 140: 'how',
 141: 'fee',
 142: 'why',
 143: 'date',
 144: 'wells',
 145: 'navient',
 146: 'accounts',
 147: 'fargo',
 148: 'times',
 149: 'funds',
 150: 'know',
 151: 'another',
 152: 'same',
 153: 'contacted',
 154: 'like',
 155: 'first',
 156: 'car',
 157: 'went',
 158: 'debt',
 159: 'than',
 160: 'charge',
 161: 'student',
 162: 'checking',
 163: 'until',
 164: 'complaint',
 165: 'issue',
 166: 'stated',
 167: 'closed',
 168: 'financial',
 169: 'want',
 170: 'charged',
 171: 'monthly',
 172: 'please',
 173: 'year',
 174: 'need',
 175: 'take',
 176: 'request',
 177: 'without',
 178: 'reporting',
 179: '2',
 180: 'see',
 181: 'going',
 182: 'several',
 183: '2016',
 184: 'before',
 185: 'able',
 186: 'business',
 187: 'name',
 188: 'however',
 189: 'charges',
 190: 'paying',
 191: 'last',
 192: 'fraud',
 193: 'work',
 194: 'online',
 195: 'statement',
 196: 'very',
 197: 'got',
 198: '3',
 199: 'representative',
 200: 'email',
 201: 'contact',
 202: 'america',
 203: 'spoke',
 204: 'process',
 205: 'go',
 206: 'well',
 207: 'receive',
 208: 'full',
 209: 'requested',
 210: 'some',
 211: 'every',
 212: 'rate',
 213: 'informed',
 214: 'send',
 215: 'dispute',
 216: 'department',
 217: 'later',
 218: 'put',
 219: 'took',
 220: 'calls',
 221: 'past',
 222: 'where',
 223: 'applied',
 224: 'address',
 225: 'file',
 226: 'each',
 227: 'deposit',
 228: 'mail',
 229: 'income',
 230: 'does',
 231: 'way',
 232: 'nothing',
 233: 'documents',
 234: 'provide',
 235: 'trying',
 236: "'m",
 237: 'branch',
 238: 'two',
 239: 'case',
 240: 'tried',
 241: 'bill',
 242: 'claim',
 243: 'making',
 244: '2015',
 245: 'today',
 246: 'under',
 247: 'stating',
 248: 'done',
 249: 'provided',
 250: 'consumer',
 251: 'reported',
 252: 'give',
 253: 'manager',
 254: 'vehicle',
 255: 'both',
 256: 'while',
 257: 'opened',
 258: 'used',
 259: "'ve",
 260: 'filed',
 261: 'insurance',
 262: 'modification',
 263: 'needed',
 264: 'someone',
 265: 'anything',
 266: 'his',
 267: 'find',
 268: 'different',
 269: 'given',
 270: 'person',
 271: 'believe',
 272: 'within',
 273: 'removed',
 274: 'him',
 275: 'many',
 276: 'current',
 277: '5',
 278: 'transaction',
 279: 'response',
 280: 'yet',
 281: 'immediately',
 282: 'refused',
 283: 'next',
 284: 'application',
 285: '1',
 286: 'notice',
 287: 'against',
 288: 'once',
 289: 'overdraft',
 290: 'order',
 291: 'use',
 292: 'problem',
 293: 'offer',
 294: 'federal',
 295: 'plan',
 296: 'already',
 297: 'matter',
 298: 'close',
 299: 'right',
 300: 'during',
 301: 'though',
 302: 'keep',
 303: 'debit',
 304: 'property',
 305: 'situation',
 306: 'signed',
 307: 'fraudulent',
 308: 'original',
 309: 'open',
 310: 'correct',
 311: 'found',
 312: 'advised',
 313: 'hold',
 314: '30',
 315: 'taken',
 316: 'feel',
 317: 'score',
 318: 'people',
 319: 'getting',
 320: 'regarding',
 321: 'its',
 322: 'additional',
 323: 'submitted',
 324: 'fact',
 325: 'denied',
 326: 'collection',
 327: 'down',
 328: 'ago',
 329: 'per',
 330: 'reason',
 331: 'having',
 332: 'copy',
 333: 'via',
 334: 'program',
 335: 'transfer',
 336: 'much',
 337: 'tell',
 338: 'explained',
 339: 'ca',
 340: 'school',
 341: 'supervisor',
 342: 'line',
 343: 'services',
 344: 'returned',
 345: 'saying',
 346: 'end',
 347: 'state',
 348: '10',
 349: 'purchase',
 350: 'such',
 351: 'repayment',
 352: 'point',
 353: 'attached',
 354: 'statements',
 355: 'house',
 356: 'proof',
 357: 'office',
 358: 'personal',
 359: 'good',
 360: 'capital',
 361: 'weeks',
 362: 'gave',
 363: 'total',
 364: 'few',
 365: 'remove',
 366: 'system',
 367: 'around',
 368: 'based',
 369: 'transactions',
 370: 'citibank',
 371: 'refund',
 372: 'set',
 373: 'cash',
 374: 'ask',
 375: 'error',
 376: 'negative',
 377: 'available',
 378: 'speak',
 379: 'documentation',
 380: 'say',
 381: 'bankruptcy',
 382: 'week',
 383: '4',
 384: 'finally',
 385: 'required',
 386: 'owe',
 387: 'unable',
 388: 'law',
 389: 'started',
 390: 'multiple',
 391: 'transferred',
 392: 'auto',
 393: 'may',
 394: 'status',
 395: 'access',
 396: 'checks',
 397: 'change',
 398: 'continue',
 399: 'agreement',
 400: 'citi',
 401: 'wanted',
 402: 'job',
 403: 'rep',
 404: 'website',
 405: 'stop',
 406: 'left',
 407: 'terms',
 408: 'asking',
 409: 'calling',
 410: 'those',
 411: 'security',
 412: 'agreed',
 413: 'wrong',
 414: 'instead',
 415: 'return',
 416: 'attorney',
 417: 'happened',
 418: 'showing',
 419: 'principal',
 420: 'thank',
 421: '15',
 422: 'equifax',
 423: 'approved',
 424: 'escrow',
 425: 'form',
 426: 'using',
 427: 'sure',
 428: 'contract',
 429: 'forbearance',
 430: 'recently',
 431: 'ever',
 432: 'dollars',
 433: 'cfpb',
 434: 'history',
 435: 'place',
 436: 'owed',
 437: 'understand',
 438: 'something',
 439: 'paperwork',
 440: 'agency',
 441: 'foreclosure',
 442: 'lender',
 443: 'sold',
 444: 'investigation',
 445: 'period',
 446: 'further',
 447: '6',
 448: 'issues',
 449: 'own',
 450: 'sale',
 451: 'cards',
 452: 'hours',
 453: 'legal',
 454: '2017',
 455: 'resolve',
 456: 'servicing',
 457: 'longer',
 458: 'nor',
 459: 'following',
 460: 'upon',
 461: 'came',
 462: 'prior',
 463: 'three',
 464: 'let',
 465: 'show',
 466: 'disputed',
 467: 'direct',
 468: 'closing',
 469: 'bonus',
 470: 'must',
 471: 'changed',
 472: 'working',
 473: 'doing',
 474: 'finance',
 475: 'private',
 476: 'customers',
 477: 'tax',
 478: 'care',
 479: 'long',
 480: 'part',
 481: 'reports',
 482: 'bureaus',
 483: 're',
 484: 'review',
 485: 'deposited',
 486: 'husband',
 487: 'note',
 488: 'everything',
 489: 'agent',
 490: 'wife',
 491: 'banking',
 492: 'letters',
 493: 'court',
 494: 'bills',
 495: 'apply',
 496: 'experian',
 497: 'offered',
 498: 'written',
 499: 'either',
 500: 'answer',
 501: 'almost',
 502: 'always',
 503: 'points',
 504: 'old',
 505: 'come',
 506: 'placed',
 507: 'states',
 508: 'atm',
 509: 'listed',
 510: 'currently',
 511: 'second',
 512: 'receiving',
 513: 'too',
 514: 'lost',
 515: 'hard',
 516: 'savings',
 517: 'failed',
 518: 'try',
 519: 'family',
 520: 'assistance',
 521: 'bureau',
 522: 'purchased',
 523: 'numerous',
 524: 'verify',
 525: 'think',
 526: '7',
 527: 'action',
 528: 'notified',
 529: 'complete',
 530: 'lower',
 531: 'writing',
 532: 'practices',
 533: 'myself',
 534: 'previous',
 535: 'american',
 536: 'most',
 537: 'act',
 538: 'identity',
 539: 'title',
 540: 'afford',
 541: 'clear',
 542: 'record',
 543: 'ocwen',
 544: 'actually',
 545: 'requesting',
 546: 'least',
 547: 'message',
 548: 'taking',
 549: 'option',
 550: 'telling',
 551: 'protection',
 552: 'allowed',
 553: 'companies',
 554: 'here',
 555: 'high',
 556: 'policy',
 557: 'look',
 558: 'default',
 559: 'incorrect',
 560: 'including',
 561: 's',
 562: 'thought',
 563: 'options',
 564: 'co',
 565: 'enough',
 566: 'records',
 567: 'agencies',
 568: 'mailed',
 569: 'dated',
 570: 'banks',
 571: 'continued',
 572: 'resolved',
 573: 'sign',
 574: 'social',
 575: 'thing',
 576: 'allow',
 577: 'completed',
 578: 'e',
 579: 'behind',
 580: 'result',
 581: 'shows',
 582: 'taxes',
 583: 'posted',
 584: "'",
 585: 'attempted',
 586: 'promotion',
 587: 'servicer',
 588: 'fair',
 589: 'noticed',
 590: 'away',
 591: 'despite',
 592: 'short',
 593: 'sending',
 594: 'pnc',
 595: 'inquiry',
 596: 'life',
 597: 'minimum',
 598: 'police',
 599: 'less',
 600: 'kept',
 601: 'issued',
 602: 'decided',
 603: 'discover',
 604: 'according',
 605: 'items',
 606: '16',
 607: 'possible',
 608: 'caused',
 609: 'aware',
 610: 'claims',
 611: 'confirmed',
 612: 'really',
 613: 'wait',
 614: 'clearly',
 615: 'added',
 616: 'boa',
 617: '100',
 618: 'theft',
 619: 't',
 620: 'says',
 621: 'explain',
 622: 'mistake',
 623: 'third',
 624: 'refuse',
 625: 'between',
 626: 'document',
 627: 'start',
 628: 'else',
 629: 'item',
 630: 'extra',
 631: '0',
 632: 'authorized',
 633: 'plus',
 634: 'checked',
 635: 'education',
 636: 'charging',
 637: 'therefore',
 638: 'representatives',
 639: 'express',
 640: 'id',
 641: 'seems',
 642: 'refinance',
 643: 'usaa',
 644: 'deferment',
 645: 'verified',
 646: 'activity',
 647: 'included',
 648: 'false',
 649: 'supposed',
 650: 'explanation',
 651: 'higher',
 652: 'payoff',
 653: 'processed',
 654: 'anyone',
 655: 'numbers',
 656: 'above',
 657: 'mother',
 658: '60',
 659: 'cover',
 660: 'began',
 661: 'twice',
 662: 'cost',
 663: 'needs',
 664: 'info',
 665: 'ally',
 666: 'delinquent',
 667: 'fax',
 668: 'missed',
 669: 'resolution',
 670: 'billing',
 671: 'merchant',
 672: 'questions',
 673: 'directly',
 674: 'collections',
 675: 'amounts',
 676: 'local',
 677: 'etc',
 678: 'filing',
 679: 'talk',
 680: '1000',
 681: 'lease',
 682: 'receipt',
 683: 'type',
 684: 'p',
 685: 'addition',
 686: 'entire',
 687: 'promised',
 688: 'paypal',
 689: 'consumers',
 690: 'college',
 691: 'limit',
 692: 'unauthorized',
 693: 'bad',
 694: 'increase',
 695: 'emails',
 696: 'illegal',
 697: 'accept',
 698: 'qualify',
 699: 'release',
 700: 'things',
 701: 'loss',
 702: 'completely',
 703: 'attempt',
 704: 'transunion',
 705: 'pending',
 706: 'practice',
 707: '12',
 708: 'along',
 709: 'responsible',
 710: 'mine',
 711: 'simply',
 712: 'nationstar',
 713: 'free',
 714: 'violation',
 715: 'purchases',
 716: '20',
 717: 'heard',
 718: 'approximately',
 719: 'minutes',
 720: 'towards',
 721: 'requirements',
 722: 'question',
 723: 'fixed',
 724: 'best',
 725: 'rates',
 726: 'problems',
 727: 'signature',
 728: '90',
 729: 'opening',
 730: 'showed',
 731: 'non',
 732: 'credited',
 733: 'follow',
 734: 'unfair',
 735: 'knew',
 736: 'settlement',
 737: '8',
 738: 'son',
 739: 'inaccurate',
 740: 'live',
 741: 'automatic',
 742: 'soon',
 743: 'submit',
 744: 'party',
 745: 'td',
 746: 'cancelled',
 747: 'held',
 748: '25',
 749: 'fix',
 750: '500',
 751: 'previously',
 752: 'creditor',
 753: 'miles',
 754: 'cancel',
 755: 'trust',
 756: 'deal',
 757: 'paper',
 758: 'inquiries',
 759: 'gone',
 760: 'buy',
 761: 'looking',
 762: 'although',
 763: 'repeatedly',
 764: 'suntrust',
 765: 'knowledge',
 766: 'waiting',
 767: 'complaints',
 768: 'hardship',
 769: 'early',
 770: 'decision',
 771: 'assured',
 772: 'collect',
 773: 'investigate',
 774: 'update',
 775: 'happen',
 776: 'notification',
 777: 'forward',
 778: 'responded',
 779: 'saw',
 780: 'employee',
 781: 'talked',
 782: 'giving',
 783: 'public',
 784: 'lien',
 785: 'dept',
 786: 'santander',
 787: 'confirmation',
 788: 'below',
 789: 'support',
 790: 'worked',
 791: 'stolen',
 792: 'morning',
 793: 'run',
 794: 'value',
 795: 'single',
 796: 'updated',
 797: 'corrected',
 798: 'better',
 799: 'synchrony',
 800: 'initial',
 801: 'wrote',
 802: '200',
 803: 'copies',
 804: 'final',
 805: 'forgiveness',
 806: 'rights',
 807: 'met',
 808: 'obtain',
 809: 'couple',
 810: 'far',
 811: 'acct',
 812: 'remaining',
 813: 'cleared',
 814: 'respond',
 815: 'indicated',
 816: 'small',
 817: 'government',
 818: 'certified',
 819: 'communication',
 820: 'conversation',
 821: 'eligible',
 822: 'spent',
 823: 'daughter',
 824: 'correspondence',
 825: 'fcra',
 826: 'deposits',
 827: 'declined',
 828: 'dates',
 829: 'occasions',
 830: 'stopped',
 831: 'dealership',
 832: 'course',
 833: 'fedloan',
 834: 'reached',
 835: 'post',
 836: 'great',
 837: 'forced',
 838: 'requests',
 839: 'claimed',
 840: 'annual',
 841: 'processing',
 842: 'union',
 843: 'u',
 844: 'lot',
 845: 'missing',
 846: 'increased',
 847: 'b',
 848: 'store',
 849: 'victim',
 850: 'code',
 851: 'ended',
 852: 'recent',
 853: 'none',
 854: 'read',
 855: 'whom',
 856: 'fault',
 857: 'willing',
 858: 'accepted',
 859: 'assist',
 860: 'bought',
 861: 'continues',
 862: 'section',
 863: 'approval',
 864: 'concern',
 865: 'confirm',
 866: 'little',
 867: 'evidence',
 868: '300',
 869: 'borrower',
 870: 'occurred',
 871: 'details',
 872: 'refuses',
 873: 'verification',
 874: 'recorded',
 875: 'real',
 876: 'means',
 877: 'telephone',
 878: 'lending',
 879: 'monday',
 880: 'contacting',
 881: 'speaking',
 882: 'became',
 883: 'rude',
 884: 'attempts',
 885: 'avoid',
 886: 'properly',
 887: 'looked',
 888: 'refunded',
 889: 'reviewed',
 890: 'causing',
 891: 'institution',
 892: 'passed',
 893: 'add',
 894: 'c',
 895: 'beginning',
 896: 'move',
 897: 'turned',
 898: 'discovered',
 899: '9',
 900: 'principle',
 901: 'dealer',
 902: 'hour',
 903: 'scheduled',
 904: 'cashed',
 905: 'low',
 906: 'reach',
 907: 'appraisal',
 908: 'wo',
 909: 'originally',
 910: '24',
 911: 'apparently',
 912: 'separate',
 913: 'regular',
 914: 'timely',
 915: 'deceptive',
 916: 'future',
 917: "'d",
 918: 'research',
 919: 'makes',
 920: 'extremely',
 921: 'deleted',
 922: 'correctly',
 923: 'outstanding',
 924: 'authorization',
 925: 'half',
 926: 'difficult',
 927: 'whole',
 928: 'true',
 929: 'accurate',
 930: 'leave',
 931: 'amex',
 932: 'lack',
 933: 'cause',
 934: 'standing',
 935: 'large',
 936: '50',
 937: '35',
 938: 'page',
 939: 'permission',
 940: 'site',
 941: 'manner',
 942: 'prove',
 943: 'unless',
 944: 'moved',
 945: 'actions',
 946: 'reversed',
 947: 'faxed',
 948: 'regards',
 949: 'emailed',
 950: 'idea',
 951: 'owner',
 952: 'employees',
 953: 'advance',
 954: 'seem',
 955: 'valid',
 956: 'advantage',
 957: 'bring',
 958: 'behalf',
 959: 'consent',
 960: 'kind',
 961: 'appears',
 962: 'inform',
 963: 'proper',
 964: 'friday',
 965: 'responsibility',
 966: 'teller',
 967: 'x',
 968: 'electronic',
 969: 'father',
 970: 'rather',
 971: 'balances',
 972: 'mentioned',
 973: 'promotional',
 974: 'providing',
 975: 'named',
 976: 'ending',
 977: 'banker',
 978: 'withdrawal',
 979: 'withdraw',
 980: 'necessary',
 981: 'lied',
 982: 'term',
 983: 'applying',
 984: 'university',
 985: 'chapter',
 986: 'thus',
 987: 'aes',
 988: 'daily',
 989: 'holder',
 990: 'automatically',
 991: 'living',
 992: 'signer',
 993: '2000',
 994: 'citizens',
 995: 'pulled',
 996: '2014',
 997: 'yes',
 998: 'equity',
 999: '120',
 1000: 'specifically',
 …}
```

1.9 Back to Decoding Our Word Vectors...

[11]:
```python
comment_idx_to_preview = 19
print('****Original complaint text:****')
print(complaints[comment_idx_to_preview])
print('\n\n')

# The reverse_index cell block above must be complete in order for this cell
# block to successively execute
decoded_review = ' '.join([reverse_index.get(i) for i in␣
 ↪sequences[comment_idx_to_preview]])
print('****Decoded review from Tokenizer:****')
print(decoded_review)
```

```
****Original complaint text:****
I have already filed several complaints about AES/PHEAA. I was notified by a
XXXX XXXX let @ XXXX, who pretended to be from your office, he said he was from
CFPB. I found out this morning he is n't from your office, but is actually works
```

```
at XXXX.

This has wasted weeks of my time. They AES/PHEAA confirmed and admitted ( see
attached transcript of XXXX, conversation at XXXX ( XXXX ) with XXXX that proves
they verified the loans are not mine ) the student loans they had XXXX, and
collected on, and reported negate credit reporting in my name are in fact, not
mine.
They conclued their investigation on XXXX admitting they made a mistake and have
my name on soneone elses loans. I these XXXX loans total {$10000.00}, original
amount. My XXXX loans I got was total {$3500.00}. We proved by providing
AES/PHEAA, this with my original promissary notes I located recently, the XXXX
of my college provided AES/PHEAA with their original shoeinf amounts of my XXXX
loans which show different dates and amounts, the dates and amounts are not even
close to matching these loans they have in my name, The original lender, XXXX
XXXX Bank notifying AES/PHEAA, they never issued me a student loan, and original
Loan Guarantor, XXXX, notifying AES/PHEAA, they never were guarantor of my
loans.

XXXX straight forward. But today, this person, XXXX XXXX, told me they know
these loans are not mine, and they refuse to remove my name off these XXXX loan
's and correct their mistake, essentially forcing me to pay these loans off,
bucause in XXXX they sold the loans to XXXX loans.

This is absurd, first protruding to be this office, and then refusing to correct
their mistake.

Please for the love of XXXX will soneone from your office call me at XXXX,
today. I am a XXXX vet and they are knowingly discriminating against me.
Pretending to be you.



****Decoded review from Tokenizer:****
i have already filed several complaints about aes i was notified by a xxxx xxxx
let xxxx who to be from your office he said he was from cfpb i found out this
morning he is n't from your office but is actually works at xxxx this has weeks
of my time they aes confirmed and admitted see attached of xxxx conversation at
xxxx xxxx with xxxx that they verified the loans are not mine the student loans
they had xxxx and on and reported credit reporting in my name are in fact not
mine they their investigation on xxxx they made a mistake and have my name on
loans i these xxxx loans total 10000 00 original amount my xxxx loans i got was
total 00 we by providing aes this with my original notes i located recently the
xxxx of my college provided aes with their original amounts of my xxxx loans
which show different dates and amounts the dates and amounts are not even close
to these loans they have in my name the original lender xxxx xxxx bank notifying
aes they never issued me a student loan and original loan xxxx notifying aes
they never were of my loans xxxx forward but today this person xxxx xxxx told me
they know these loans are not mine and they refuse to remove my name off these
```

```
xxxx loan 's and correct their mistake essentially me to pay these loans off in
xxxx they sold the loans to xxxx loans this is first to be this office and then
refusing to correct their mistake please for the of xxxx will from your office
call me at xxxx today i am a xxxx and they are against me to be you
```

1.10 Convert the Products to Numerical Categories

On to step two of our preprocessing: converting our descriptive categories into integers.

[18]:
```
product = df['Product']

# Initialize
le = preprocessing.LabelEncoder()
le.fit(product)
print('****Original class labels:****')
print(list(le.classes_))
print('\n')
product_cat = le.transform(product)

# If you wish to retrieve the original descriptive labels post production
# list(le.inverse_transform([0, 1, 3, 3, 0, 6, 4]))

print('****New product labels:****')
print(product_cat)
print('\n')

# Each row will be all zeros except for the category for that observation
print('****One hot labels; 7 binary columns, one for each of the categories.
 ↪****')
product_onehot = to_categorical(product_cat)
print(product_onehot)
print('\n')

print('****One hot labels shape:****')
print(np.shape(product_onehot))
```

```
****Original class labels:****
['Bank account or service', 'Checking or savings account', 'Consumer Loan',
'Credit card', 'Credit reporting', 'Mortgage', 'Student loan']



****New product labels:****
[6 6 6 … 4 4 4]



****One hot labels; 7 binary columns, one for each of the categories.****
[[0. 0. 0. … 0. 0. 1.]
 [0. 0. 0. … 0. 0. 1.]
 [0. 0. 0. … 0. 0. 1.]
 …
 [0. 0. 0. … 1. 0. 0.]
 [0. 0. 0. … 1. 0. 0.]
 [0. 0. 0. … 1. 0. 0.]]


****One hot labels shape:****
(60000, 7)
```

1.11 Train-test split

Now for our final preprocessing step: the usual train-test split.

[19]:
```
random.seed(123)
test_index = random.sample(range(1,10000), 1500)

test = one_hot_results[test_index]
train = np.delete(one_hot_results, test_index, 0)

label_test = product_onehot[test_index]
label_train = np.delete(product_onehot, test_index, 0)

print('Test label shape:', np.shape(label_test))
print('Train label shape:', np.shape(label_train))
print('Test shape:', np.shape(test))
print('Train shape:', np.shape(train))
```

```
Test label shape: (1500, 7)
Train label shape: (58500, 7)
Test shape: (1500, 2000)
Train shape: (58500, 2000)
```

1.12 Building the network

Let's build a fully connected (Dense) layer network with relu activation in Keras. You can do this using: `Dense(16, activation='relu')`.

In this example, use two hidden layers with 50 units in the first layer and 25 in the second, both with a `'relu'` activation function. Because we are dealing with a multiclass problem (classifying the complaints into 7 categories), we use a use a `'softmax'` classifier in order to output 7 class probabilities per case.

[24]:
```
# Initialize a sequential model
model = models.Sequential()

# Two layers with relu activation
model.add(layers.Dense(50, activation = "relu", input_shape =(2000,)))
model.add(layers.Dense(25, activation = "relu"))
```

```
# One layer with softmax activation
model.add(layers.Dense(7, activation = "softmax"))
```

1.13 Compiling the model

Now, compile the model! This time, use `'categorical_crossentropy'` as the loss function and stochastic gradient descent, `'SGD'` as the optimizer. As in the previous lesson, include the accuracy as a metric.

[25]:
```
# Compile the model
model.compile(optimizer = optimizers.SGD(lr = 0.001),
             loss = "categorical_crossentropy",
             metrics = ["accuracy"])
```

1.14 Training the model

In the compiler, you'll be passing the optimizer (SGD = stochastic gradient descent), loss function, and metrics. Train the model for 120 epochs in mini-batches of 256 samples.

Note: Your code may take about one to two minutes to run.

[26]:
```
# Train the model
history = model.fit(train, label_train,
                   epochs = 120, batch_size = 256,
                    validation_data = (test, label_test))
```

```
Epoch 1/120
229/229 [==============================] - 1s 3ms/step - loss: 1.9402 -
accuracy: 0.1650 - val_loss: 1.9028 - val_accuracy: 0.1667
Epoch 2/120
229/229 [==============================] - 1s 3ms/step - loss: 1.9256 -
accuracy: 0.1824 - val_loss: 1.8739 - val_accuracy: 0.2153
Epoch 3/120
229/229 [==============================] - 1s 3ms/step - loss: 1.9128 -
accuracy: 0.2009 - val_loss: 1.8437 - val_accuracy: 0.2600
Epoch 4/120
229/229 [==============================] - 1s 3ms/step - loss: 1.9003 -
accuracy: 0.2203 - val_loss: 1.8100 - val_accuracy: 0.3160
Epoch 5/120
229/229 [==============================] - 1s 3ms/step - loss: 1.8872 -
accuracy: 0.2414 - val_loss: 1.7749 - val_accuracy: 0.3847
Epoch 6/120
229/229 [==============================] - 1s 3ms/step - loss: 1.8731 -
accuracy: 0.2645 - val_loss: 1.7375 - val_accuracy: 0.4520
Epoch 7/120
229/229 [==============================] - 1s 3ms/step - loss: 1.8577 -
```

```
accuracy: 0.2881 - val_loss: 1.6942 - val_accuracy: 0.5287
Epoch 8/120
229/229 [==============================] - 1s 3ms/step - loss: 1.8407 -
accuracy: 0.3116 - val_loss: 1.6522 - val_accuracy: 0.5847
Epoch 9/120
229/229 [==============================] - 1s 3ms/step - loss: 1.8219 -
accuracy: 0.3369 - val_loss: 1.6073 - val_accuracy: 0.6380
Epoch 10/120
229/229 [==============================] - 1s 3ms/step - loss: 1.8012 -
accuracy: 0.3614 - val_loss: 1.5654 - val_accuracy: 0.6827
Epoch 11/120
229/229 [==============================] - 1s 3ms/step - loss: 1.7786 -
accuracy: 0.3877 - val_loss: 1.5200 - val_accuracy: 0.7207
Epoch 12/120
229/229 [==============================] - 1s 3ms/step - loss: 1.7542 -
accuracy: 0.4121 - val_loss: 1.4776 - val_accuracy: 0.7407
Epoch 13/120
229/229 [==============================] - 1s 3ms/step - loss: 1.7278 -
accuracy: 0.4368 - val_loss: 1.4342 - val_accuracy: 0.7647
Epoch 14/120
229/229 [==============================] - 1s 3ms/step - loss: 1.6995 -
accuracy: 0.4604 - val_loss: 1.3918 - val_accuracy: 0.7740
Epoch 15/120
229/229 [==============================] - 1s 3ms/step - loss: 1.6690 -
accuracy: 0.4816 - val_loss: 1.3471 - val_accuracy: 0.7973
Epoch 16/120
229/229 [==============================] - 1s 3ms/step - loss: 1.6362 -
accuracy: 0.5043 - val_loss: 1.2984 - val_accuracy: 0.8107
Epoch 17/120
229/229 [==============================] - 1s 3ms/step - loss: 1.6010 -
accuracy: 0.5234 - val_loss: 1.2518 - val_accuracy: 0.8227
Epoch 18/120
229/229 [==============================] - 1s 3ms/step - loss: 1.5632 -
accuracy: 0.5401 - val_loss: 1.2071 - val_accuracy: 0.8307
Epoch 19/120
229/229 [==============================] - 1s 3ms/step - loss: 1.5230 -
accuracy: 0.5551 - val_loss: 1.1632 - val_accuracy: 0.8393
Epoch 20/120
229/229 [==============================] - 1s 3ms/step - loss: 1.4818 -
accuracy: 0.5688 - val_loss: 1.1168 - val_accuracy: 0.8427
Epoch 21/120
229/229 [==============================] - 1s 3ms/step - loss: 1.4405 -
accuracy: 0.5817 - val_loss: 1.0736 - val_accuracy: 0.8413
Epoch 22/120
229/229 [==============================] - 1s 3ms/step - loss: 1.3995 -
accuracy: 0.5946 - val_loss: 1.0292 - val_accuracy: 0.8427
Epoch 23/120
229/229 [==============================] - 1s 3ms/step - loss: 1.3591 -
```

```
accuracy: 0.6055 - val_loss: 0.9819 - val_accuracy: 0.8507
Epoch 24/120
229/229 [==============================] - 1s 3ms/step - loss: 1.3194 -
accuracy: 0.6162 - val_loss: 0.9460 - val_accuracy: 0.8507
Epoch 25/120
229/229 [==============================] - 1s 3ms/step - loss: 1.2809 -
accuracy: 0.6260 - val_loss: 0.9064 - val_accuracy: 0.8553
Epoch 26/120
229/229 [==============================] - 1s 3ms/step - loss: 1.2437 -
accuracy: 0.6361 - val_loss: 0.8645 - val_accuracy: 0.8567
Epoch 27/120
229/229 [==============================] - 1s 2ms/step - loss: 1.2079 -
accuracy: 0.6445 - val_loss: 0.8303 - val_accuracy: 0.8613
Epoch 28/120
229/229 [==============================] - 1s 3ms/step - loss: 1.1738 -
accuracy: 0.6515 - val_loss: 0.7959 - val_accuracy: 0.8627
Epoch 29/120
229/229 [==============================] - 1s 3ms/step - loss: 1.1414 -
accuracy: 0.6582 - val_loss: 0.7666 - val_accuracy: 0.8647
Epoch 30/120
229/229 [==============================] - 1s 3ms/step - loss: 1.1107 -
accuracy: 0.6646 - val_loss: 0.7360 - val_accuracy: 0.8667
Epoch 31/120
229/229 [==============================] - 1s 3ms/step - loss: 1.0819 -
accuracy: 0.6708 - val_loss: 0.7094 - val_accuracy: 0.8693
Epoch 32/120
229/229 [==============================] - 1s 3ms/step - loss: 1.0547 -
accuracy: 0.6762 - val_loss: 0.6846 - val_accuracy: 0.8713
Epoch 33/120
229/229 [==============================] - 1s 3ms/step - loss: 1.0293 -
accuracy: 0.6812 - val_loss: 0.6759 - val_accuracy: 0.8627
Epoch 34/120
229/229 [==============================] - 1s 3ms/step - loss: 1.0055 -
accuracy: 0.6861 - val_loss: 0.6415 - val_accuracy: 0.8713
Epoch 35/120
229/229 [==============================] - 1s 3ms/step - loss: 0.9833 -
accuracy: 0.6897 - val_loss: 0.6249 - val_accuracy: 0.8687
Epoch 36/120
229/229 [==============================] - 1s 3ms/step - loss: 0.9626 -
accuracy: 0.6935 - val_loss: 0.6142 - val_accuracy: 0.8700
Epoch 37/120
229/229 [==============================] - 1s 3ms/step - loss: 0.9433 -
accuracy: 0.6970 - val_loss: 0.5932 - val_accuracy: 0.8720
Epoch 38/120
229/229 [==============================] - 1s 3ms/step - loss: 0.9252 -
accuracy: 0.7007 - val_loss: 0.5816 - val_accuracy: 0.8707
Epoch 39/120
229/229 [==============================] - 1s 3ms/step - loss: 0.9083 -
```

```
accuracy: 0.7034 - val_loss: 0.5636 - val_accuracy: 0.8740
Epoch 40/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8926 -
accuracy: 0.7070 - val_loss: 0.5528 - val_accuracy: 0.8727
Epoch 41/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8779 -
accuracy: 0.7099 - val_loss: 0.5379 - val_accuracy: 0.8740
Epoch 42/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8642 -
accuracy: 0.7125 - val_loss: 0.5289 - val_accuracy: 0.8760
Epoch 43/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8513 -
accuracy: 0.7147 - val_loss: 0.5185 - val_accuracy: 0.8753
Epoch 44/120
229/229 [==============================] - 1s 2ms/step - loss: 0.8392 -
accuracy: 0.7174 - val_loss: 0.5068 - val_accuracy: 0.8780
Epoch 45/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8278 -
accuracy: 0.7195 - val_loss: 0.4880 - val_accuracy: 0.8800
Epoch 46/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8172 -
accuracy: 0.7222 - val_loss: 0.4934 - val_accuracy: 0.8740
Epoch 47/120
229/229 [==============================] - 1s 3ms/step - loss: 0.8071 -
accuracy: 0.7238 - val_loss: 0.4833 - val_accuracy: 0.8753
Epoch 48/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7977 -
accuracy: 0.7259 - val_loss: 0.4731 - val_accuracy: 0.8773
Epoch 49/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7887 -
accuracy: 0.7279 - val_loss: 0.4662 - val_accuracy: 0.8760
Epoch 50/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7803 -
accuracy: 0.7303 - val_loss: 0.4569 - val_accuracy: 0.8780
Epoch 51/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7723 -
accuracy: 0.7312 - val_loss: 0.4540 - val_accuracy: 0.8767
Epoch 52/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7647 -
accuracy: 0.7333 - val_loss: 0.4470 - val_accuracy: 0.8780
Epoch 53/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7576 -
accuracy: 0.7349 - val_loss: 0.4421 - val_accuracy: 0.8787
Epoch 54/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7507 -
accuracy: 0.7359 - val_loss: 0.4431 - val_accuracy: 0.8753
Epoch 55/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7442 -
```

```
accuracy: 0.7381 - val_loss: 0.4307 - val_accuracy: 0.8780
Epoch 56/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7380 -
accuracy: 0.7397 - val_loss: 0.4251 - val_accuracy: 0.8793
Epoch 57/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7321 -
accuracy: 0.7417 - val_loss: 0.4216 - val_accuracy: 0.8760
Epoch 58/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7265 -
accuracy: 0.7425 - val_loss: 0.4104 - val_accuracy: 0.8840
Epoch 59/120
229/229 [==============================] - 1s 3ms/step - loss: 0.7211 -
accuracy: 0.7437 - val_loss: 0.4045 - val_accuracy: 0.8847
Epoch 60/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7159 -
accuracy: 0.7453 - val_loss: 0.4064 - val_accuracy: 0.8820
Epoch 61/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7109 -
accuracy: 0.7468 - val_loss: 0.4029 - val_accuracy: 0.8827
Epoch 62/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7062 -
accuracy: 0.7481 - val_loss: 0.3918 - val_accuracy: 0.8860
Epoch 63/120
229/229 [==============================] - 1s 2ms/step - loss: 0.7016 -
accuracy: 0.7493 - val_loss: 0.3965 - val_accuracy: 0.8847
Epoch 64/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6972 -
accuracy: 0.7509 - val_loss: 0.3819 - val_accuracy: 0.8913
Epoch 65/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6930 -
accuracy: 0.7518 - val_loss: 0.3831 - val_accuracy: 0.8887
Epoch 66/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6889 -
accuracy: 0.7531 - val_loss: 0.3825 - val_accuracy: 0.8900
Epoch 67/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6849 -
accuracy: 0.7544 - val_loss: 0.3778 - val_accuracy: 0.8913
Epoch 68/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6812 -
accuracy: 0.7559 - val_loss: 0.3757 - val_accuracy: 0.8913
Epoch 69/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6775 -
accuracy: 0.7568 - val_loss: 0.3648 - val_accuracy: 0.8953
Epoch 70/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6740 -
accuracy: 0.7579 - val_loss: 0.3679 - val_accuracy: 0.8920
Epoch 71/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6705 -
```

```
accuracy: 0.7584 - val_loss: 0.3630 - val_accuracy: 0.8933
Epoch 72/120
229/229 [==============================] - 1s 3ms/step - loss: 0.6672 -
accuracy: 0.7603 - val_loss: 0.3604 - val_accuracy: 0.8953
Epoch 73/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6640 -
accuracy: 0.7613 - val_loss: 0.3593 - val_accuracy: 0.8947
Epoch 74/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6609 -
accuracy: 0.7619 - val_loss: 0.3555 - val_accuracy: 0.8960
Epoch 75/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6579 -
accuracy: 0.7630 - val_loss: 0.3531 - val_accuracy: 0.8960
Epoch 76/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6549 -
accuracy: 0.7635 - val_loss: 0.3508 - val_accuracy: 0.8953
Epoch 77/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6520 -
accuracy: 0.7651 - val_loss: 0.3486 - val_accuracy: 0.8953
Epoch 78/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6493 -
accuracy: 0.7656 - val_loss: 0.3476 - val_accuracy: 0.8940
Epoch 79/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6465 -
accuracy: 0.7666 - val_loss: 0.3422 - val_accuracy: 0.8973
Epoch 80/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6439 -
accuracy: 0.7680 - val_loss: 0.3459 - val_accuracy: 0.8933
Epoch 81/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6413 -
accuracy: 0.7681 - val_loss: 0.3451 - val_accuracy: 0.8913
Epoch 82/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6388 -
accuracy: 0.7691 - val_loss: 0.3344 - val_accuracy: 0.8953
Epoch 83/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6363 -
accuracy: 0.7698 - val_loss: 0.3382 - val_accuracy: 0.8947
Epoch 84/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6339 -
accuracy: 0.7705 - val_loss: 0.3324 - val_accuracy: 0.8973
Epoch 85/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6316 -
accuracy: 0.7718 - val_loss: 0.3290 - val_accuracy: 0.8987
Epoch 86/120
229/229 [==============================] - 1s 3ms/step - loss: 0.6293 -
accuracy: 0.7722 - val_loss: 0.3330 - val_accuracy: 0.8973
Epoch 87/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6271 -
```

```
accuracy: 0.7729 - val_loss: 0.3311 - val_accuracy: 0.8973
Epoch 88/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6249 -
accuracy: 0.7733 - val_loss: 0.3244 - val_accuracy: 0.9013
Epoch 89/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6228 -
accuracy: 0.7745 - val_loss: 0.3302 - val_accuracy: 0.8973
Epoch 90/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6206 -
accuracy: 0.7752 - val_loss: 0.3222 - val_accuracy: 0.9007
Epoch 91/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6186 -
accuracy: 0.7757 - val_loss: 0.3267 - val_accuracy: 0.8980
Epoch 92/120
229/229 [==============================] - 1s 3ms/step - loss: 0.6166 -
accuracy: 0.7763 - val_loss: 0.3166 - val_accuracy: 0.9020
Epoch 93/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6147 -
accuracy: 0.7770 - val_loss: 0.3226 - val_accuracy: 0.8980
Epoch 94/120
229/229 [==============================] - 1s 3ms/step - loss: 0.6127 -
accuracy: 0.7775 - val_loss: 0.3222 - val_accuracy: 0.8980
Epoch 95/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6108 -
accuracy: 0.7781 - val_loss: 0.3166 - val_accuracy: 0.8987
Epoch 96/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6089 -
accuracy: 0.7791 - val_loss: 0.3219 - val_accuracy: 0.8980
Epoch 97/120
229/229 [==============================] - 1s 3ms/step - loss: 0.6071 -
accuracy: 0.7796 - val_loss: 0.3155 - val_accuracy: 0.8987
Epoch 98/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6053 -
accuracy: 0.7800 - val_loss: 0.3168 - val_accuracy: 0.8987
Epoch 99/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6036 -
accuracy: 0.7812 - val_loss: 0.3141 - val_accuracy: 0.8987
Epoch 100/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6018 -
accuracy: 0.7817 - val_loss: 0.3147 - val_accuracy: 0.9000
Epoch 101/120
229/229 [==============================] - 1s 2ms/step - loss: 0.6001 -
accuracy: 0.7822 - val_loss: 0.3117 - val_accuracy: 0.9007
Epoch 102/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5985 -
accuracy: 0.7831 - val_loss: 0.3127 - val_accuracy: 0.9000
Epoch 103/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5968 -
```

```
accuracy: 0.7835 - val_loss: 0.3091 - val_accuracy: 0.9000
Epoch 104/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5952 -
accuracy: 0.7844 - val_loss: 0.3056 - val_accuracy: 0.9020
Epoch 105/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5936 -
accuracy: 0.7848 - val_loss: 0.3063 - val_accuracy: 0.9027
Epoch 106/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5920 -
accuracy: 0.7854 - val_loss: 0.3046 - val_accuracy: 0.9027
Epoch 107/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5905 -
accuracy: 0.7857 - val_loss: 0.3055 - val_accuracy: 0.9027
Epoch 108/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5890 -
accuracy: 0.7871 - val_loss: 0.3058 - val_accuracy: 0.9033
Epoch 109/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5875 -
accuracy: 0.7874 - val_loss: 0.3002 - val_accuracy: 0.9060
Epoch 110/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5861 -
accuracy: 0.7880 - val_loss: 0.3006 - val_accuracy: 0.9060
Epoch 111/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5846 -
accuracy: 0.7888 - val_loss: 0.3025 - val_accuracy: 0.9040
Epoch 112/120
229/229 [==============================] - 1s 2ms/step - loss: 0.5832 -
accuracy: 0.7889 - val_loss: 0.2986 - val_accuracy: 0.9067
Epoch 113/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5818 -
accuracy: 0.7898 - val_loss: 0.2994 - val_accuracy: 0.9067
Epoch 114/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5804 -
accuracy: 0.7902 - val_loss: 0.2966 - val_accuracy: 0.9073
Epoch 115/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5790 -
accuracy: 0.7908 - val_loss: 0.2959 - val_accuracy: 0.9073
Epoch 116/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5777 -
accuracy: 0.7909 - val_loss: 0.2989 - val_accuracy: 0.9060
Epoch 117/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5763 -
accuracy: 0.7916 - val_loss: 0.2982 - val_accuracy: 0.9067
Epoch 118/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5750 -
accuracy: 0.7925 - val_loss: 0.2954 - val_accuracy: 0.9073
Epoch 119/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5737 -
```

```
accuracy: 0.7926 - val_loss: 0.2936 - val_accuracy: 0.9080
Epoch 120/120
229/229 [==============================] - 1s 3ms/step - loss: 0.5725 -
accuracy: 0.7929 - val_loss: 0.2906 - val_accuracy: 0.9087
```

Recall that the dictionary **history** has two entries: the loss and the accuracy achieved using the training set.

[27]:
```
history_dict = history.history
history_dict.keys()
```

[27]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[30]:
```
len(history_dict["loss"])
```

[30]:
```
120
```

[31]:
```
history_dict["loss"]
```

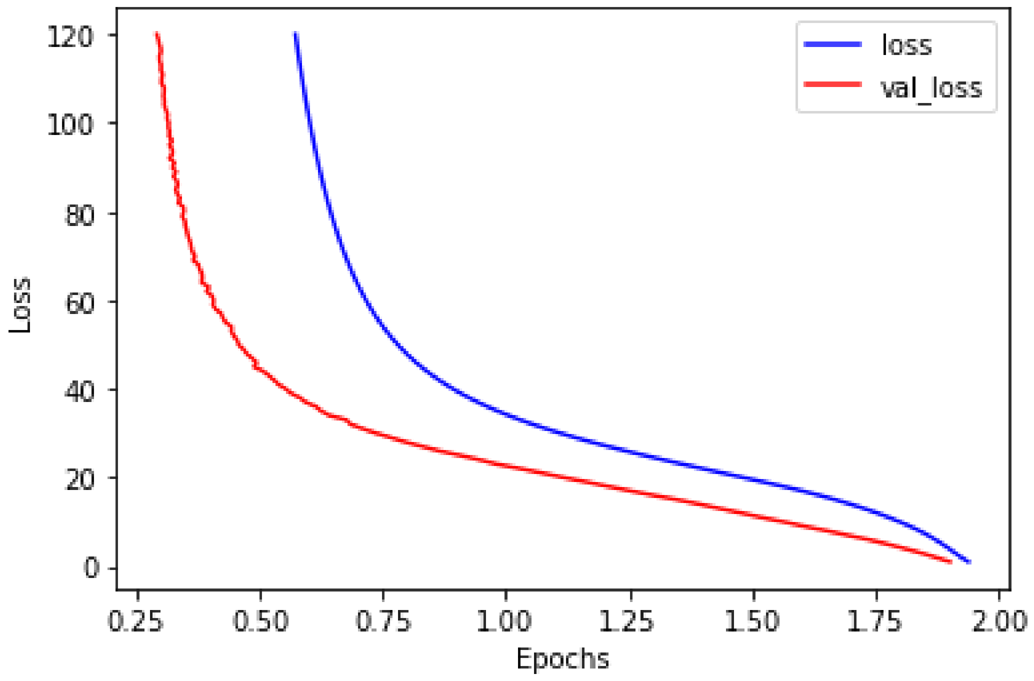
[31]:
```
[1.940170407295227,
 1.925602912902832,
 1.9128127098083496,
 1.9002768993377686,
 1.8872315883636475,
 1.87314772605896,
 1.8577134609222412,
 1.8407161235809326,
 1.821906328201294,
 1.801216721534729,
 1.7786372900009155,
 1.7542058229446411,
 1.7278482913970947,
 1.6994588375091553,
 1.6689502000808716,
 1.6362205743789673,
 1.6010154485702515,
 1.5631777048110962,
 1.5229747295379639,
 1.4817780256271362,
 1.440524697303772,
 1.39952552318573,
 1.3590641021728516,
 1.3194193840026855,
 1.280882477760315,
 1.2436662912368774,
 1.2079159021377563,
 1.1738029718399048,
 1.141403079032898,
 1.1107484102249146,
 1.0818766355514526,
 1.0547116994857788,
 1.0293012857437134,
 1.005528211593628,
 0.9833424091339111,
 0.9626079201698303,
 0.9432525634765625,
 0.9251869916915894,
 0.9083443880081177,
 0.8926182985305786,
 0.8779101967811584,
 0.8641621470451355,
 0.8512691259384155,
 0.8392069339752197,
 0.8278313279151917,
 0.8171550035476685,
 0.8071215748786926,
 0.7976893782615662,
 0.7887409925460815,
 0.7802894115447998,
 0.7723367810249329,
 0.7646954655647278,
 0.7575507760047913,
 0.7506799101829529,
 0.7442086935043335,
 0.7380111217498779,
 0.732083261013031,
 0.7264513969421387,
 0.7210618257522583,
 0.7159001231193542,
 0.7109441161155701,
 0.7061807513237,
 0.7016211152076721,
 0.6972104907035828,
 0.6930088996887207,
 0.6889354586601257,
 0.6849357485771179,
 0.6812071204185486,
 0.6775378584861755,
 0.6739890575408936,
 0.6705486178398132,
 0.6672499179840088,
 0.6640186309814453,
 0.660884439945221,
 0.6578512191772461,
 0.6548885703086853,
 0.6520082950592041,
 0.6492584347724915,
 0.6465248465538025,
 0.6438798308372498,
 0.6413002014160156,
 0.6387823224067688,
 0.6363223195075989,
 0.6339440941810608,
 0.631584644317627,
 0.6292991638183594,
 0.627086877822876,
 0.6248899698257446,
 0.622765839099884,
 0.6206396818161011,
 0.618618905544281,
 0.6165679693222046,
 0.6146518588066101,
 0.6126982569694519,
 0.6107912063598633,
 0.6089293360710144,
 0.6071038246154785,
 0.6052884459495544,
 0.6035743355751038,
 0.6018334627151489,
 0.6001359224319458,
 0.5984547138214111,
 0.5968042612075806,
 0.5951976776123047,
 0.5935918688774109,
 0.5920450687408447,
 0.5905095338821411,
 0.5890340805053711,
 0.5874968767166138,
 0.586054265499115,
 0.5846068859100342,
 0.5831601023674011,
 0.5817565321922302,
 0.5803568959236145,
 0.5790221691131592,
 0.5776734948158264,
 0.5763276815414429,
 0.5749931931495667,
 0.5737032294273376,
 0.5724509358406067]
```

1.15 Plot the results

As you might expect, we'll use our `matplotlib` for graphing. Use the data stored in the `history_dict` above to plot the loss vs epochs and the accuracy vs epochs.

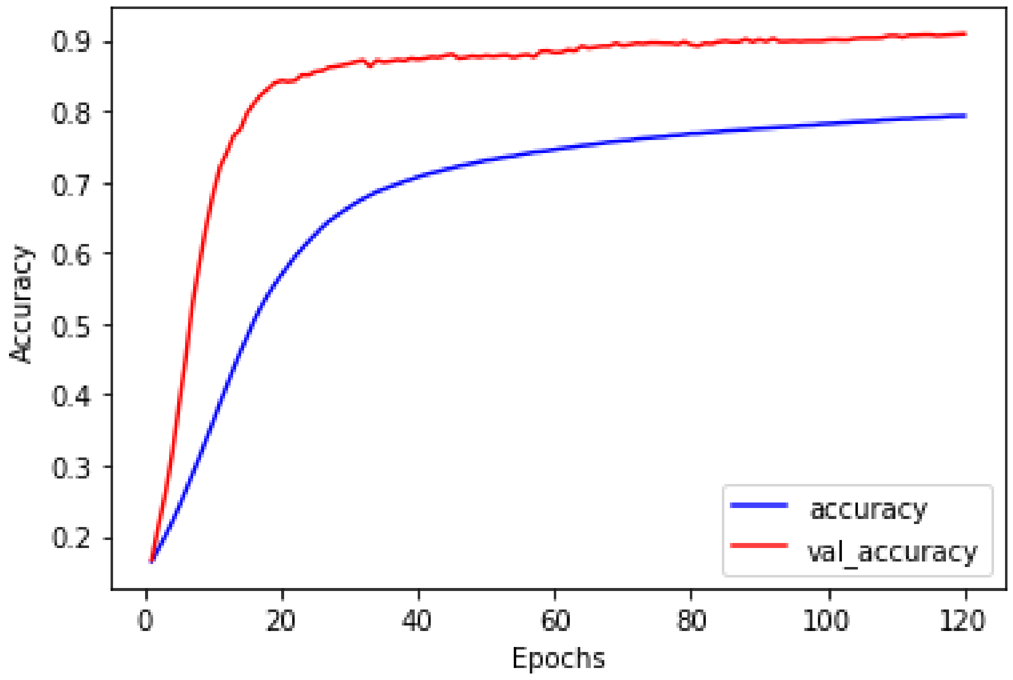
[45]:
```
# Plot the loss vs the number of epoch
plt.plot(history_dict["loss"], range(1, len(history_dict["loss"])+1),
         color = "blue", label = "loss");
plt.plot(history_dict["val_loss"], range(1, len(history_dict["loss"])+1),
         color = "red", label = "val_loss");

# plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend();
```



[47]:
```
# Plot the training accuracy vs the number of epochs
plt.plot(range(1, len(history_dict["loss"])+1), history_dict["accuracy"],
         color = "blue", label = "accuracy");
plt.plot(range(1, len(history_dict["loss"])+1), history_dict["val_accuracy"],
         color = "red", label = "val_accuracy");

plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend();
```



It seems like we could just keep on going and accuracy would go up!

1.16 Make predictions

Finally, it's time to make predictions. Use the relevant method discussed in the previous lesson to output (probability) predictions for the test set.

[44]:
```
# Output (probability) predictions for the test set
y_hat_test = model.predict(test)
```

1.17 Evaluate Performance

Finally, print the loss and accuracy for both the train and test sets of the final trained model.

[48]:
```
# Print the loss and accuracy for the training set
results_train = model.evaluate(train, label_train)
results_train
```

```
1829/1829 [==============================] - 1s 760us/step - loss: 0.5715 -
accuracy: 0.7938
```

[48]:
```
[0.5714541673660278, 0.7937948703765869]
```

[49]:
```
# Print the loss and accuracy for the test set
results_test = model.evaluate(test, label_test)
results_test
```

```
47/47 [==============================] - 0s 801us/step - loss: 0.2906 -
accuracy: 0.9087
```

[49]:
```
[0.2906302213668823, 0.9086666703224182]
```

We can see that the training set results are really good, and the test set results seem to be even better. In general, this type of result will be rare, as train set results are usually at least a bit better than test set results.

1.18 Additional Resources

- https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Consumer_complaints.ipynb
- https://catalog.data.gov/dataset/consumer-complaint-database

1.19 Summary

Congratulations! In this lab, you built a neural network thanks to the tools provided by Keras! In upcoming lessons and labs we'll continue to investigate further ideas regarding how to tune and refine these models for increased accuracy and performance.