Import modules ¶

SUMMARY

Data was scraped from seek.com.au and the following datapoints were extracted in relation to job posted on the website: category Job Category company Company placing the Ad jobclass Type of job - Data Science, Data Analyst and Business Intelligence jobdate Data Job posted jobid Job Id jobtext Free text on the jobsite location Job location eg: Melbourne subcategory Job Subcategory suburb Job suburb eg: Melbourne CBD title Job Title worktype Work Type eg: Full Time

The following tasks are performed:

1. Clean up the free text like remove punctuations
2. Create train and test datasets
3. Feature for the models - jobtext (free text) and Label - Title
4. Fit training dataset to CountVectoriser
5. Run Gridsearch to get best Ngram paremeter
6. Run 12 classification models on train dataset and get accuracy scores to enable choose the best model
7. Train the model with the best accuracy score
8. Run the model with Test dataset
9. Check accuracy of the model

In [2]:

```
import requests
from bs4 import BeautifulSoup
import lxml
import numpy as np
import pandas as pd
from scipy.stats import norm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import re
# Getting that SKLearn Dataset
from sklearn.datasets import fetch_20newsgroups
%matplotlib inline

# machine learning
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB

# Clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import normalize
from sklearn.metrics import silhouette_score

#Gridsearch and scoring
from sklearn.grid_search import GridSearchCV
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import accuracy_score

# Train test
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import HashingVectorizer, TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score
```

```
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\cross_validatio
n.py:41: DeprecationWarning: This module was deprecated in version 0.18 in f
avor of the model_selection module into which all the refactored classes and
functions are moved. Also note that the interface of the new CV iterators ar
e different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\grid_search.py:4
2: DeprecationWarning: This module was deprecated in version 0.18 in favor o
f the model_selection module into which all the refactored classes and funct
```

```
ions are moved. This module will be removed in 0.20.
  DeprecationWarning)
```

Load data

In [3]:

```
dfe = pd.read_csv('./Data/jobsdf.csv', index_col=0)
```

In [4]:

```
dfe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1768 entries, 0 to 1767
Data columns (total 15 columns):
category       1768 non-null object
company        1746 non-null object
jobclass       1768 non-null object
jobdate        1764 non-null object
jobid          1768 non-null int64
jobtext        1744 non-null object
location       1768 non-null object
subcategory    1768 non-null object
suburb         1296 non-null object
title          1768 non-null object
worktype       1764 non-null object
period         1768 non-null object
uppersal       289 non-null float64
lowersal       388 non-null float64
finalsal       388 non-null float64
dtypes: float64(3), int64(1), object(11)
memory usage: 221.0+ KB
```

In [5]:

```
# Classify salary
sal_type = []
for i in dfe.finalsal:
    if i > 0:
        sal_type.append(1)
    else:
        sal_type.append(0)
dfe['saltype'] = sal_type
```

In [6]:

```
# Cleanup data
import string
dfe['jobtext'].replace(np.nan,'NA',inplace=True)
dfe['jobtext'].replace('\n','',inplace=True)
dfe['jobtext'] = dfe['jobtext'].apply(lambda x:''.join([i for i in x
                                                  if i not in string.punctuation]))
dfe['jobtext'].str.replace(r'/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/','')
```

Out[6]:

```
0        Offices in the USA UK and Australia Predictiv...
1         About SEEK SEEK is a diverse group of compan...
2        Associate Data Scientist Applied Machine Lear...
3        Data Scientist Applied Machine Learning About...
4        Ignite Data Solutions are a data and analytic...
5        Our client is a national reputable and wellkn...
6       We are seeking up to 2 Senior data scientists ...
7       Our client is an industry pioneering Customer ...
8         Do you want work in a place where  Your supe...
9        Overview This multinational FinTech startup b...
10      Our client is one of Brisbanes fastest growing...
11         About the business and the role My client i...
12       Are you curious about how digital is changing...
13      Our client is one of Brisbanes leading Big Dat...
14       Offices in the USA UK and Australia Predictiv...
15       Are you a passionate Data Scientist looking t...
16         Data Scientist Our client an innovative sta...
17         Data Scientist  We are currently recruiting...
18      My client is a marketleading IT ServicesSI who...
19       The organization is seeking a highly experien...
20      We are looking for a Data Scientist to join th...
21       We are a world leading Health  Care organisat...
22      My client is a marketleading IT ServicesSI who...
23      Researcher  Data Scientist AI Machine Learning...
24       About the Company One of Australia’s leading ...
25       Data3 Limited DTL is an ASX listed company th...
26       Required skills    A minimum of 5 years exper...
27       The Company   My client is a wellestablished ...
28       A leading consulting company is driving a tra...
29      My client is a marketleading IT ServicesSI who...
                              ...
1738      Whats it like working in the Engineering tea...
1739      FIRESOFT Consulting is a multiaward winning ...
1740     Our client is the largest and fastest growing...
1741      Melbourne PFM Head Office Supportive team en...
1742         Our Client is a global software vendor th...
1743     Looking to speak with the highest calibre of ...
1744     The role Our client a household Australian na...
1745     This is a precursor to improving the experien...
1746     Technology Solutions Manager     My client op...
1747     This is a precursor to improving the experien...
1748     We are looking for an agile astute and affabl...
1749     My client a well known enterprise Retail orga...
1750        Global Company Market Leader Be part of a ...
1751     This is a precursor to improving the experien...
1752     Since 2008 Servian has been known to be one o...
1753     Servian was established in 2008 and has consi...
1754     Since 2008 Servian has been known to be one o...
1755      Represent one of the true cloudbased ERP sol...
```

```
1756     Progressive in partnership with one of the wo...
1757     Bluefin Resources are working with a prestigi...
1758     Based in Crows Nest acidgreen are an multiawa...
1759      Major project great team environment Permane...
1760     Excellent Corporate Package  Super  Company C...
1761     Are you a talented and skilled professional l...
1762     ABOUT USGAPbuster Worldwide is a global leade...
1763     We are currently recruiting for a Data Analys...
1764     Junior Sales Executive - Corporate Event Indu...
1765     The financial markets business of this global...
1766     Leading and managing an offshore and onshore ...
1767     South Eastern Sydney Local Health District SE...
Name: jobtext, Length: 1768, dtype: object
```

In [7]:

```
# Train and Test datasets FOR TITLES (SALARY TYPE)
X = dfe['jobtext']
#y=dfe.title
y=dfe.saltype
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=101)
```

In [8]:

```
# # Establish baseline
base_line = np.mean(dfe.finalsal)
base_line
```

Out[8]:

```
129809.76417525773
```

In [9]:

```
# Fit training dataset
vect= CountVectorizer(stop_words='english', ngram_range=(1,2))
vect.fit(X_train)
```

Out[9]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 2), preprocessor=None, stop_words='english',
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [10]:

```
# Create DataFrame with vectorised words
XX_train = pd.DataFrame(vect.transform(X_train).todense(), columns= vect.get_feature_names(
```

In [11]:

```
# Check top 20 words
word_counts = XX_train.sum(axis=0)
word_counts.sort_values(ascending = False).head(20)
```

Out[11]:

```
data            7531
experience      4597
business        4384
skills          2377
role            2214
team            2132
work            1830
management      1639
working         1603
strong          1435
development     1389
analysis        1268
ability         1261
reporting       1259
information     1241
analytics       1228
apply           1202
support         1152
intelligence    1106
solutions       1058
dtype: int64
```

In [12]:

```
# Initialise NLTK word tokenizer
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
class LemmaTokenizer(object):
    def __init__(self):
        self.wnl = WordNetLemmatizer()
    def __call__(self, articles):
        return [self.wnl.lemmatize(t) for t in word_tokenize(articles)]
```

Run Gridsearch to get the best parameter for Ngram Range

In [ ]:

```
# Run Gridsearch to get the best parameter for Ngram Range
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('mult', LogisticRegression()),
])
CV_parameters = { 'vect__ngram_range':((1, 1),(1, 2), (2, 2), (1, 3)),
                  'vect__stop_words': ['english'],
}
CV_gridsearch = GridSearchCV(pipeline, CV_parameters, n_jobs=-1, verbose=1)

CV_gridsearch.fit(X_train, y_train)
print ('Gridsearch Best Score: ', CV_gridsearch.best_score_,'\n')
print ('Gridsearch Best Parameters: \n', CV_gridsearch.best_params_,'\n')
predictions = CV_gridsearch.predict(X_test)
print('Accuracy Score: ', accuracy_score(y_test, predictions),'\n')
print('Classification Report: \n', classification_report(predictions,y_test),'\n')
print('Confusion Matrix: \n', confusion_matrix(predictions,y_test),'\n')
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits

C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\model_selection
\_split.py:605: Warning: The least populated class in y has only 1 members,
which is too few. The minimum number of members in any class cannot be less
than n_splits=3.
  % (min_groups, self.n_splits)), Warning)
```

In [23]:

```
# Check gridsearch output parameters
# print ('Gridsearch Best Score: ', CV_gridsearch.best_score_,'\n')
# print ('Gridsearch Best Parameters: \n', CV_gridsearch.best_params_,'\n')
# predictions = CV_gridsearch.predict(X_test)
# print('Accuracy Score: ', accuracy_score(y_test, predictions),'\n')
# print('Classification Report: \n', classification_report(predictions,y_test),'\n')
# print('Confusion Matrix: \n', confusion_matrix(predictions,y_test),'\n')
```

Out[23]:

```
<function __main__.run_Gridsearch>
```

Run 12 classification models on train dataset and get accuracy scores to choose the best model

In [122]:

```
#Creates a function to try the different models and print out the score
from sklearn.tree import DecisionTreeClassifier
def Models_Acc(XTrain, yTrain, XTest, yTest, ngramRange=(1, 2)):
    for i in range(1,12):
        if i == 1:
            model = make_pipeline(HashingVectorizer(stop_words='english',
                                                    non_negative=True,
                                                    n_features=2**16),
                    LogisticRegression())

            print ('-------------------------------------------------------------------
            print ('Hashing Vectorizer and Logistic Regression')

        elif i == 2:
            model = make_pipeline(TfidfVectorizer(stop_words='english',
                                        sublinear_tf=True,
                                        max_df=0.5,
                                        max_features=1000,
                                        ngram_range=ngramRange),
                    LogisticRegression())

            print ('-------------------------------------------------------------------
            print ('TfidfVectorizer using',format(ngramRange), 'ngram and Logistic Regressi
        elif i == 3:

            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                          LogisticRegression()
                          )
            print ('-------------------------------------------------------------------
            print ('CountVectorizer with', format(ngramRange), 'ngram and Logistic Regressi

        elif i == 4:
            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                          LogisticRegression()
                          )
            print ('-------------------------------------------------------------------
            print ('CountVectorizer with LemmaTokenizer and ', format(ngramRange), 'ngram a

        elif i == 5:
            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                          DecisionTreeClassifier(),
                      )
            print ('-------------------------------------------------------------------
            print ('CountVectorizer with ', format(ngramRange), ' and MultinomialNB Regress

        elif i == 6:
            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                          RandomForestClassifier(),
                      )
            print ('-------------------------------------------------------------------
            print ('CountVectorizer with ', format(ngramRange), ' and Random Forest Regress
        elif i == 7:

            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                          DecisionTreeClassifier())
            print ('-------------------------------------------------------------------
            print ('CountVectorizer with ',format(ngramRange), 'and Decision Tree')

        elif i == 8:
            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                                  AdaBoostClassifier())
            print ('------------------------------------------------------------------------
            print ('CountVectorizer with ',format(ngramRange), ' and AdaBoost')
        elif i == 9:

            model = make_pipeline(TfidfVectorizer(stop_words='english',
                                  sublinear_tf=True,
                                  max_df=0.5,
                                  max_features=1000,
                                  ngram_range=ngramRange),
                        DecisionTreeClassifier())

            print ('------------------------------------------------------------------------
            print ('TfidfVectorizer using ',format(ngramRange),' ngram and DecisionTreeClas

        elif i == 10:
            model = make_pipeline(TfidfVectorizer(stop_words='english',
                                  sublinear_tf=True,
                                  max_df=0.5,
                                  max_features=1000,
                                  ngram_range=ngramRange),
                        RandomForestClassifier())

            print ('------------------------------------------------------------------------
            print ('TfidfVectorizer using ',format(ngramRange) ,'ngram and RandomForestClas

        elif i ==11:
            model = make_pipeline(CountVectorizer(stop_words='english', ngram_range=ngramRa
                                          SVC(kernel='linear'))
            print ('------------------------------------------------------------------------
            print ('CountVectorizer with LemmaTokenizer with ',format(ngramRange),' and SVM

        model.fit(XTrain, yTrain)
        y_pred = model.predict(XTest)
        print ('Accuracy:',accuracy_score(yTest, y_pred))
```

In [123]:

```
# Check accuracy of models
Models_Acc(X_train, y_train, X_test, y_test, ngramRange=(2, 2))
```

```
----------------------------------------------------------------------------
-------------
Hashing Vectorizer and Logistic Regression

C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\feature_extracti
on\hashing.py:94: DeprecationWarning: the option non_negative=True has been
deprecated in 0.19 and will be removed in version 0.21.
  " in version 0.21.", DeprecationWarning)
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\feature_extracti
on\hashing.py:94: DeprecationWarning: the option non_negative=True has been
deprecated in 0.19 and will be removed in version 0.21.
  " in version 0.21.", DeprecationWarning)
C:\Users\Vinita Auplish\Anaconda3\lib\site-packages\sklearn\feature_extracti
on\hashing.py:94: DeprecationWarning: the option non_negative=True has been
deprecated in 0.19 and will be removed in version 0.21.
  " in version 0.21.", DeprecationWarning)

Accuracy: 0.7740112994350282
----------------------------------------------------------------------------
-------------
TfidfVectorizer using (2, 2) ngram and Logistic Regression
Accuracy: 0.7871939736346516
----------------------------------------------------------------------------
-------------
CountVectorizer with (2, 2) ngram and Logistic Regression
Accuracy: 0.839924670433145
----------------------------------------------------------------------------
-------------
CountVectorizer with LemmaTokenizer and  (2, 2) ngram and Logistic Regressio
n
Accuracy: 0.8361581920903954
----------------------------------------------------------------------------
-------------
CountVectorizer with  (2, 2)  and MultinomialNB Regression
Accuracy: 0.8022598870056498
----------------------------------------------------------------------------
-------------
CountVectorizer with  (2, 2)  and Random Forest Regression
Accuracy: 0.8361581920903954
----------------------------------------------------------------------------
-------------
CountVectorizer with  (2, 2) and Decision Tree
Accuracy: 0.7966101694915254
----------------------------------------------------------------------------
-------------
CountVectorizer with  (2, 2)  and AdaBoost
Accuracy: 0.775894538606403
----------------------------------------------------------------------------
-------------
TfidfVectorizer using  (2, 2)  ngram and DecisionTreeClassifier
Accuracy: 0.7532956685499058
----------------------------------------------------------------------------
-------------
TfidfVectorizer using  (2, 2) ngram and RandomForestClassifier
Accuracy: 0.8173258003766478
----------------------------------------------------------------------------
```

```
--------------
CountVectorizer with LemmaTokenizer with  (2, 2)  and SVM
Accuracy: 0.8305084745762712
```

Run Logistic Regression for Title (Salary Type) as that has given best accuracy score

In [13]:

```
# Lets use the stop_words argument to remove words like "and, the, a"
cvec = CountVectorizer(stop_words='english', ngram_range=(2,2))

# Fit our vectorizer using our train data
cvec.fit(X_train)
# Write vectorised words back to the dataframe
X_train = pd.DataFrame(cvec.transform(X_train).todense(),
                       columns=cvec.get_feature_names())
X_test = pd.DataFrame(cvec.transform(X_test).todense(),
                      columns=cvec.get_feature_names())
# RANDOM FOREST TAKES TOO LONG
# Import the model we are using
# from sklearn.ensemble import RandomForestRegressor
# # Instantiate model with 1000 decision trees
# rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
# rf.fit(X_train, y_train)
# rf.score(X_test, y_test)
#Import and fit our logistic regression and test it too
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
```

Out[13]:

```
0.839924670433145
```

In [14]:

```
# Generate predictions from Logistic Regression
predictions = lr.predict(X_test)
print('Confusion Matrix: \n', confusion_matrix(y_test,predictions),'\n')
print('Classification Report: \n', classification_report(y_test,predictions),'\n')
```

```
Confusion Matrix:
 [[399  11]
 [ 74  47]]

Classification Report:
              precision    recall  f1-score   support

          0       0.84      0.97      0.90       410
          1       0.81      0.39      0.53       121

avg / total       0.84      0.84      0.82       531
```

In [15]:

```
# List top 30 predicted values from Logistic Regression
columns  =  np.array(cvec.get_feature_names())
def list_important_pred_var(data):
    important_pred_var = pd.DataFrame(data, columns = ["coef"], index = columns)
    return important_pred_var.sort_values(["coef"], ascending = False)[:30]

list_important_pred_var(lr.coef_.T)
```

Out[15]:

	coef
experience strong	0.365296
data engineer	0.336160
like apply	0.323484
confidence danieletegroupcomau	0.321953
danieletegroupcomau hot	0.321953
data modelling	0.315801
financial services	0.300634
send resume	0.299526
apply button	0.286014
initial month	0.284226
purpose position	0.275545
finance business	0.259355
development experience	0.257839
apply updated	0.245693
workforce planning	0.243020
sydney cbd	0.236132
analytics reporting	0.226177
based sydney	0.225985
work closely	0.225204
relevant experience	0.223625
internal stakeholders	0.222484
ability interpret	0.217470
ad hoc	0.214430
business partner	0.212060
apply feel	0.209412
data engineering	0.209228
high level	0.207831
data entry	0.203614
experience needed	0.203610
site business	0.196369

In [16]:

```
predictions
```

Out[16]:

```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0], dtype=int64)
```

In [17]:

```
predictions.sum()
```

Out[17]:

```
58
```

In [ ]:

```

```