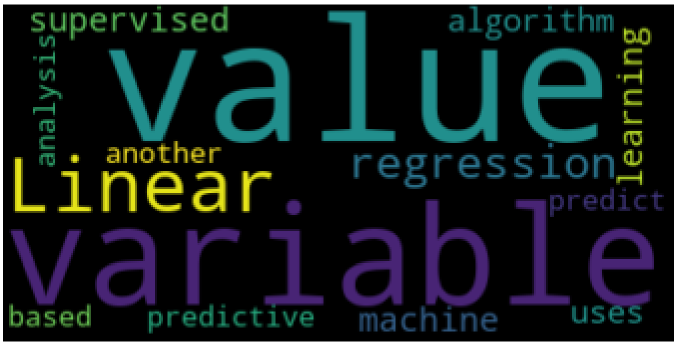
In [ ]:

```
#import required packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

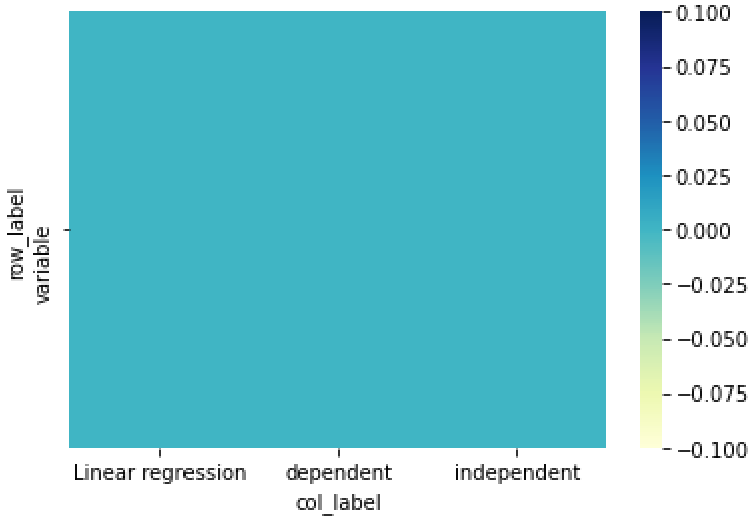
In [15]:

```
from wordcloud import WordCloud

text = '''Linear regression is a supervised machine learning algorithm
          ,uses predictive analysis to predict the value of variable based
           on value of another variable  '''

wordcloud = WordCloud().generate(text)

import matplotlib.pyplot as plt
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



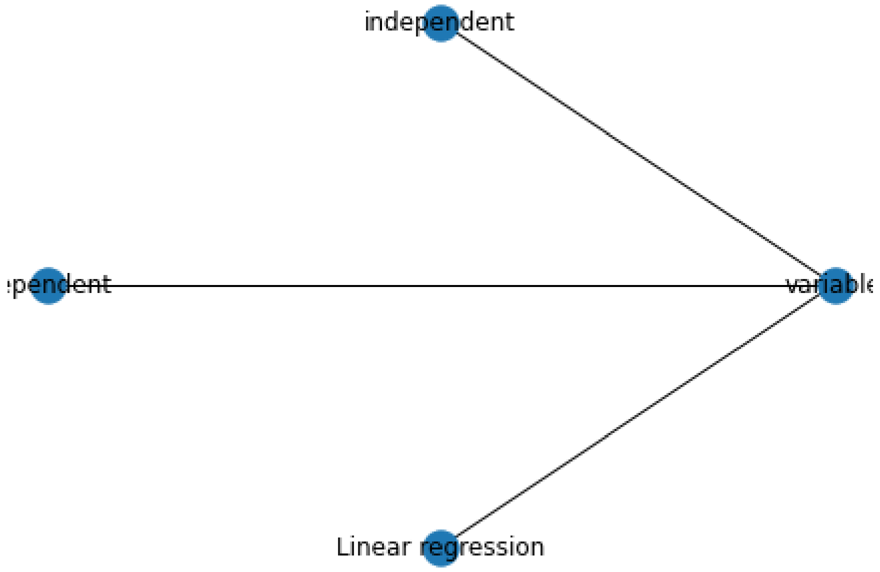
In [16]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# define the text and word pairs
text = ''' Linear regression uses predictive analysis to predict
            the value of dependent variable based
            on value of independent variable'''

word_pairs = [("variable", "independent"), ("variable", "dependent"), ("variable", "Linear

# count the frequency of each word pair
freq_table = {"_".join(pair): text.count(pair[0] + " " + pair[1])
              for pair in word_pairs}

# create a heatmap using the frequency table
heatmap_data = pd.DataFrame.from_dict(freq_table, orient='index')
heatmap_data.columns = ['count']
heatmap_data['row_label'], heatmap_data['col_label'] = zip(*heatmap_data.index.str.split('_
#heatmap_data = heatmap_data.pivot_table(index="row_label", columns="col_label", values="co
heatmap_data = heatmap_data.pivot_table(index="row_label", columns="col_label", values="cou
sns.heatmap(heatmap_data, cmap="YlGnBu")

# show the plot
plt.show()
```



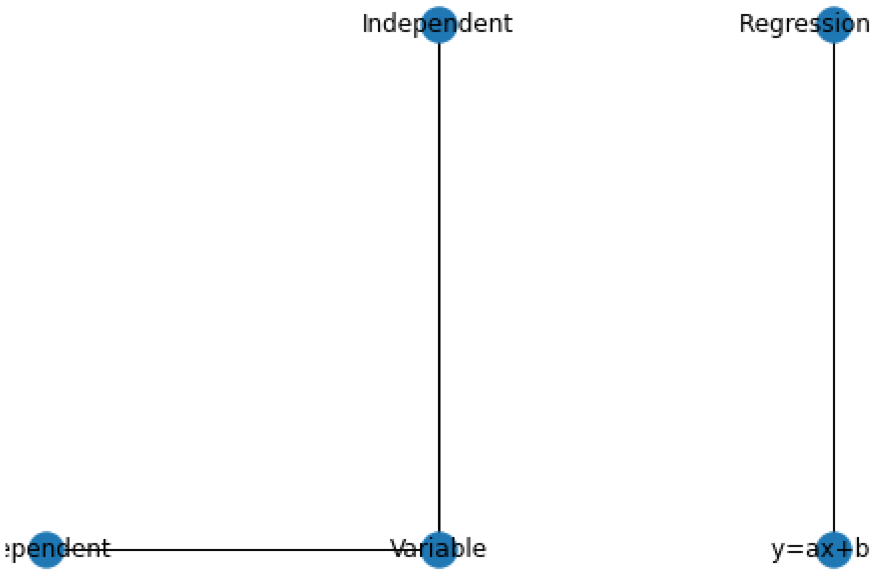
In [17]:

```
import networkx as nx
import matplotlib.pyplot as plt

word_pairs = [("variable", "independent"), ("variable", "dependent"),
              ("variable", "Linear regression")]


G = nx.Graph()
# add edges to the graph
for pair in word_pairs:
    G.add_edge(pair[0], pair[1])

# set the node positions in a circular layout
pos = nx.circular_layout(G)

# draw the graph
nx.draw(G, with_labels=True, pos=pos)

# show the plot
plt.show()
```



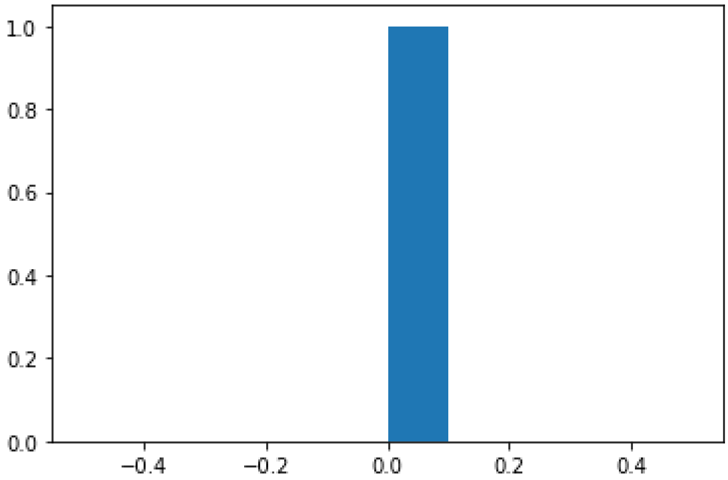
In [22]:

```
import networkx as nx
import matplotlib.pyplot as plt

# create a graph
G = nx.Graph()

# add nodes to the graph
G.add_node("dependent")
G.add_node("variable")
G.add_node("independent")
G.add_node("equ")
G.add_node("reg")

# add edges to the graph
G.add_edge("dependent", "variable")
G.add_edge("variable", "independent")
G.add_edge("equ", "reg")

# set node positions
pos = {
    "dependent": (0, 0),
    "variable": (1, 0),
    "independent": (1, 1),
    "equ": (2, 0),
    "reg": (2, 1)
}

# set node labels
labels = {
    "dependent": "Dependent",
    "variable": "Variable",
    "independent": "Independent",
    "equ": "y=ax+b   ",
    "reg": "Regression       "
}

# draw the graph with labels and positions
nx.draw(G, pos=pos, with_labels=True, labels=labels)

# show the plot
plt.show()
```



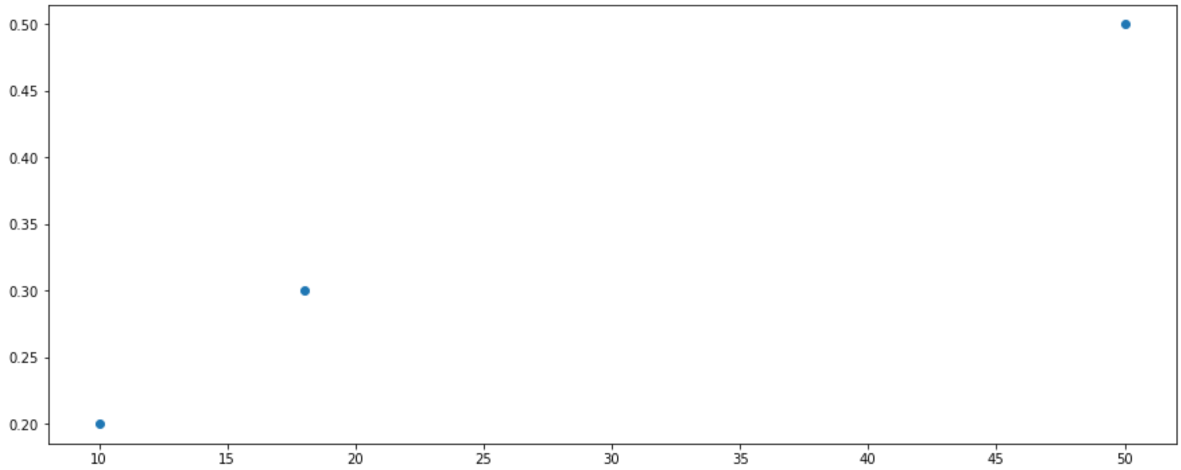
In [23]:

```
from textblob import TextBlob
import matplotlib.pyplot as plt

text = ''' Linear regression uses predictive analysis to predict
            the value of dependent variable based
            on value of independent variable'''
blob = TextBlob(text)
sentiment_scores = [sentence.sentiment.polarity for sentence in blob.sentences]

plt.hist(sentiment_scores)
plt.show()
```



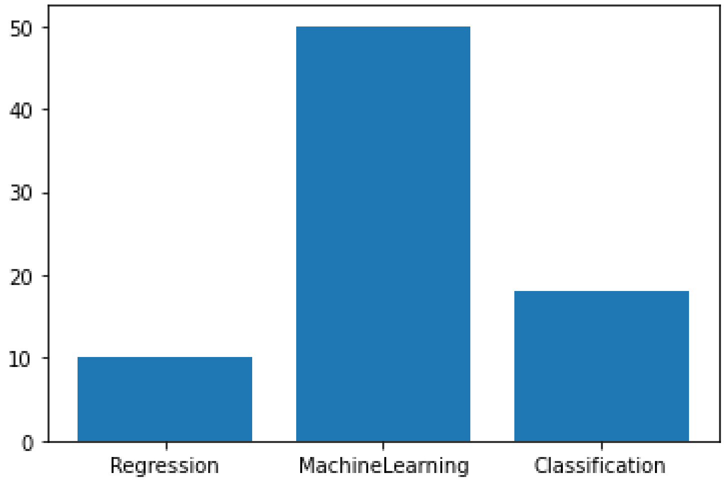
In [27]:

```
import matplotlib.pyplot as plt

# create a scatterplot of word frequency vs. sentiment score
freq_data = {"Regression": 10, "MachineLearning": 50, "Classification": 18}
sentiment_data = {"Regression": 0.2, "MachineLearning": 0.5,
                  "Classification": 0.3}
plt.scatter(list(freq_data.values()), list(sentiment_data.values()))
plt.show()
```



In [24]:

```
import matplotlib.pyplot as plt

# create a bar chart of word frequency
freq_data = {"Regression": 10, "MachineLearning": 50, "Classification": 18}
plt.bar(list(freq_data.keys()), list(freq_data.values()))
plt.show()
```



In [25]:

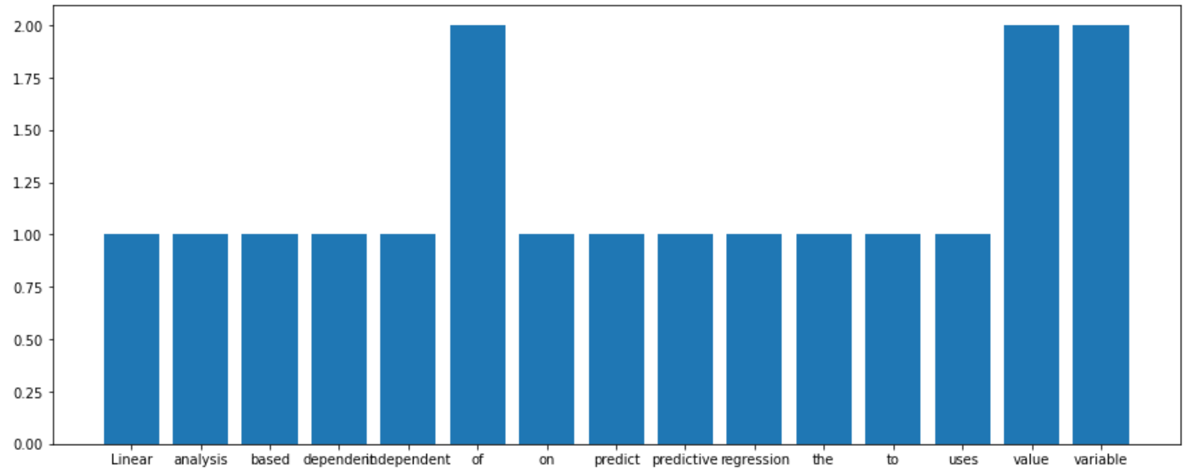
```
textn = ["Linear regression uses predictive analysis",
         " to predict the value of dependent variable",
         "based  on value of independent variable"]
td=""
for i in textn:
    td=td+i+" "
tdd=td.split()
tdd_word=sorted(set(tdd))
tdd_word
wordcount=dict()
for x in tdd_word:

    wordcount[x]=tdd.count(x)
wordcount
```

Out[25]:

```
{'Linear': 1,
 'analysis': 1,
 'based': 1,
 'dependent': 1,
 'independent': 1,
 'of': 2,
 'on': 1,
 'predict': 1,
 'predictive': 1,
 'regression': 1,
 'the': 1,
 'to': 1,
 'uses': 1,
 'value': 2,
 'variable': 2}
```

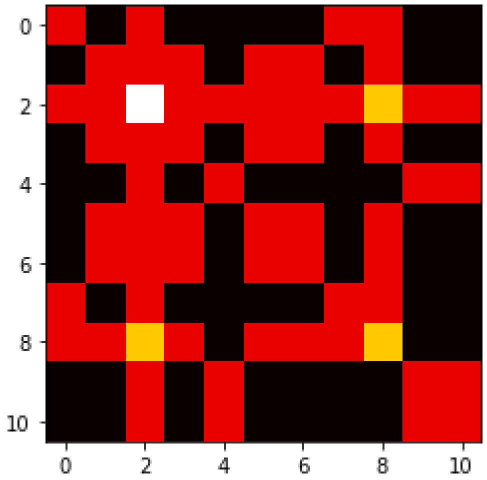
In [26]:

```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (15, 6)
plt.rcParams['font.size'] = 10
#plt.xlabel('X Axis Label', rotation=0)
#plt.xlabel('X Axis Label', fontsize=18)
plt.bar(list(wordcount.keys()), list(wordcount.values()))

plt.show()
```



In [35]:

```
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt

text = ["example text for co-occurrence matrix",
        "another example of text",
        "this is a third example"]

# create a co-occurrence matrix
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text)
cooc_matrix = (X.T * X).toarray()

# plot the matrix
plt.imshow(cooc_matrix, cmap='hot', interpolation='nearest')
plt.show()
```



In [ ]:

```

```