In [6]:

```
import nltk
import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn import model_selection, naive_bayes, metrics, svm
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.neural_network import MLPClassifier
```

In [ ]:

```
preprocessing, linear_model,
from sklearn import decomposition, ensemble
```

Base de dados

In [2]:

```
def open_file(filepath):
    file = open(filepath, 'r')

    return file.read()
```

In [3]:

```
senhora = open_file('books/senhora.txt')
diva = open_file('books/diva.txt')
gaucho = open_file('books/gaucho.txt')
guarani = open_file('books/guarani.txt')
iracema = open_file('books/iracema.txt')
luciola = open_file('books/luciola.txt')
viuvinha = open_file('books/viuvinha.txt')
ubirajara = open_file('books/ubirajara.txt')
sertanejo = open_file('books/sertanejo.txt')
troncodoipe = open_file('books/troncodoipe.txt')
```

In [4]:

```
# Lista com os textos crus
texts = [senhora, diva, gaucho, guarani, iracema, luciola, viuvinha, ubirajara,
sertanejo, troncodoipe]
```

In [5]:

```
labels = ['u', 'u', 'r', 'i', 'i', 'u', 'u', 'i', 'r', 'r']
```

In [7]:

```
df = pd.DataFrame()
df['texts'] = texts
df['labels'] = labels
df
```

Out[7]:

	texts	labels
0	Há anos raiou no céu fluminense uma nova estre...	u
1	Emília tinha quatorze anos quando a vi pela pr...	u
2	Como são melancólicas e solenes, ao pino do so...	r
3	De um dos cabeços da Serra dos Órgãos desliza ...	i
4	Verdes mares bravios de minha terra natal, ond...	i
5	A senhora estranhou, na última vez que estivem...	u
6	Se passasse há dez anos pela Praia da Glória, ...	u
7	Pela marjem do grande rio caminha Jaguarê, o j...	i
8	Esta imensa campina, que se dilata por horizon...	r
9	Era linda a situação da fazenda de Nossa Senho...	r

In [8]:

```
BAD_SYMBOLS_RE = re.compile('[^ \w]')
STOPWORDS = set(stopwords.words('portuguese'))
```

In [9]:

```
def preProcess(text):
    text = text.lower()
    text = ' '.join(word for word in text.split() if word not in STOPWORDS)
    text = BAD_SYMBOLS_RE.sub(' ', text)

    return text
```

In [10]:

```
df['texts'] = df['texts'].apply(preProcess)
df
```

Out[10]:

	texts	labels
0	anos raiou céu fluminense nova estrela desde ...	u
1	emília quatorze anos vi primeira vez menina f...	u
2	melancólicas solenes pino sol vastas campina...	r
3	cabeços serra órgãos desliza fio água dirige n...	i
4	verdes mares bravios terra natal onde canta j...	i
5	senhora estranhou última vez juntos excessiv...	u
6	passasse dez anos praia glória prima antes n...	u
7	marjem grande rio caminha jaguarê joven caçad...	i
8	imensa campina dilata horizontes infindos é ...	r
9	linda situação fazenda senhora boqueirão agua...	r

Vetores

In [11]:

```
count_vectorizer = CountVectorizer(ngram_range=(2,2))      # Count_vector
tf_idf_vectorizer = TfidfVectorizer(ngram_range=(2,2))     # TF-IDF normalizado
vectorizer = TfidfVectorizer(ngram_range=(2,2), norm=None)  #TF-IDF sem normaliz
ação
```

In [12]:

```
X1 = count_vectorizer.fit_transform(df['texts'])
X2 = tf_idf_vectorizer.fit_transform(df['texts'])
X3 = vectorizer.fit_transform(df['texts'])
```

In [13]:

```
X1.toarray()
```

Out[13]:

```
array([[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, 1],
       [3, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [14]:

```
X2.toarray()
```

Out[14]:

```
array([[0.        , 0.00442859, 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.
   ,
         0.00201503],
       [0.01030604, 0.        , 0.        , ..., 0.        , 0.
   ,
         0.        ]])
```

In [15]:

```
X3.toarray()
```

Out[15]:

```
array([[0.        , 2.70474809, 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.
   ,
         2.70474809],
       [6.89784895, 0.        , 0.        , ..., 0.        , 0.
   ,
         0.        ]])
```

Classificação com Scikit-learn

In [16]:

```
classifiers = [
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, n_iter=5, random_state=
42),
    DecisionTreeClassifier(max_depth=5),
    MultinomialNB(),
    GaussianNB(),
    MLPClassifier(alpha=1)]
```

In [44]:

```
x_train, x_test, y_train, y_test = model_selection.train_test_split(df['texts'],
 df['labels'])
```

In [45]:

```
X_train = tf_idf_vectorizer.fit_transform(x_train)
```

In [46]:

```
X_test = tf_idf_vectorizer.transform(x_test)
```

In [47]:

```
def classification(clf, X_train, X_test):
    c = clf.fit(X_train, y_train)

    return c.predict(X_test)
```

In [48]:

```
svm_linear = classification(classifiers[0], X_train, X_test)
svm = classification(classifiers[1], X_train, X_test)
svm_sgdc = classification(classifiers[2], X_train, X_test)
decision_tree = classification(classifiers[3], X_train, X_test)
multi_naive = classification(classifiers[4], X_train, X_test)
gauss_naive = classification(classifiers[5], X_train.toarray(), X_test.toarray
())
neural_net = classification(classifiers[6], X_train, X_test)
```

```
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/linear_mod
el/stochastic_gradient.py:117: DeprecationWarning: n_iter parameter
is deprecated in 0.19 and will be removed in 0.21. Use max_iter and
tol instead.
  DeprecationWarning)
```

In [49]:

```
print(y_test)
print("------------------------------------")
print("Linear SVM")
print(svm_linear, np.mean(svm_linear == y_test))
print("RBF SVM")
print(svm, np.mean(svm == y_test))
print("SGDC SVM")
print(svm_sgdc, np.mean(svm_sgdc == y_test))
print("Decision Tree")
print(decision_tree, np.mean(decision_tree == y_test))
print("Multinominal Naive")
print(multi_naive, np.mean(multi_naive == y_test))
print("Gaussian Naive Bayes")
print(gauss_naive, np.mean(gauss_naive == y_test))
print("Neural Net")
print(neural_net, np.mean(neural_net == y_test))
```

```
4    i
3    i
1    u
Name: labels, dtype: object
------------------------------------
Linear SVM
['r' 'u' 'u'] 0.3333333333333333
RBF SVM
['r' 'u' 'u'] 0.3333333333333333
SGDC SVM
['i' 'u' 'u'] 0.6666666666666666
Decision Tree
['u' 'u' 'u'] 0.3333333333333333
Multinominal Naive
['r' 'u' 'u'] 0.3333333333333333
Gaussian Naive Bayes
['r' 'r' 'u'] 0.3333333333333333
Neural Net
['u' 'u' 'u'] 0.3333333333333333
```

In [50]:

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import classification_report
```

In [51]:

```
def pipeline(classifier):
    pipe = Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', classifier)])

    return pipe
```

In [52]:

```
my_tags = ['u', 'r', 'i']
```

In [53]:

```
svm_linear = pipeline(classifiers[0])
svm_linear.fit(x_train, y_train)
svm_linear_pred = svm_linear.predict(x_test)

print("Linear SVM")
print(f"real classes: {y_test.values}")
print(f"predicted classes: {svm_linear_pred}")
print(f"accuracy: {metrics.accuracy_score(svm_linear_pred, y_test)}")
print(classification_report(y_test, svm_linear_pred, target_names = my_tags))
```

```
real classes: ['i' 'i' 'u']
predicted classes: ['r' 'u' 'u']
accuracy: 0.3333333333333333
             precision    recall  f1-score   support

          u       0.00      0.00      0.00         2
          r       0.00      0.00      0.00         0
          i       0.50      1.00      0.67         1

avg / total       0.17      0.33      0.22         3


/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted sam
ples.
  'precision', 'predicted', average, warn_for)
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are
ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

In [59]:

```
svm = pipeline(classifiers[1])
svm.fit(x_train, y_train)
svm_pred = svm.predict(x_test)

print("RBF SVM")
print(f"real classes: {y_test.values}")
print(f"predicted classes: {svm_pred}")
print(f"accuracy: {metrics.accuracy_score(svm_pred, y_test)}")
print(classification_report(y_test, svm_pred, target_names = my_tags))
```

```
RBF SVM
real classes: ['i' 'i' 'u']
predicted classes: ['r' 'u' 'u']
accuracy: 0.3333333333333333
             precision    recall  f1-score   support

          u       0.00      0.00      0.00         2
          r       0.00      0.00      0.00         0
          i       0.50      1.00      0.67         1

avg / total       0.17      0.33      0.22         3


/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted sam
ples.
  'precision', 'predicted', average, warn_for)
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are
ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

In [60]:

```
svm_sgdc = pipeline(classifiers[2])
svm_sgdc.fit(x_train, y_train)
svm_sgdc_pred = svm_sgdc.predict(x_test)

print("SGDC SVM")
print(f"real classes: {y_test.values}")
print(f"predicted classes: {svm_sgdc_pred}")
print(f"accuracy: {metrics.accuracy_score(svm_sgdc_pred, y_test)}")
print(classification_report(y_test, svm_sgdc_pred, target_names = my_tags))
```

```
SGDC SVM
real classes: ['i' 'i' 'u']
predicted classes: ['i' 'u' 'u']
accuracy: 0.6666666666666666
             precision    recall  f1-score   support

          u       1.00      0.50      0.67         2
          r       0.50      1.00      0.67         1

avg / total       0.83      0.67      0.67         3


/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/linear_mod
el/stochastic_gradient.py:117: DeprecationWarning: n_iter parameter
is deprecated in 0.19 and will be removed in 0.21. Use max_iter and
tol instead.
  DeprecationWarning)
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1428: UserWarning: labels size, 2, does not match si
ze of target_names, 3
  .format(len(labels), len(target_names))
```

In [61]:

```
decision_tree = pipeline(classifiers[3])
decision_tree.fit(x_train, y_train)
decision_tree_pred = decision_tree.predict(x_test)

print("Decision Tree")
print(f"real classes: {y_test.values}")
print(f"predicted classes: {decision_tree_pred}")
print(f"accuracy: {metrics.accuracy_score(decision_tree_pred, y_test)}")
print(classification_report(y_test, decision_tree_pred, target_names = my_tags))
```

```
Decision Tree
real classes: ['i' 'i' 'u']
predicted classes: ['u' 'r' 'r']
accuracy: 0.0
             precision    recall  f1-score   support

          u       0.00      0.00      0.00         2
          r       0.00      0.00      0.00         0
          i       0.00      0.00      0.00         1

avg / total       0.00      0.00      0.00         3


/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted sam
ples.
  'precision', 'predicted', average, warn_for)
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are
ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

In [65]:

```
multi_naive = pipeline(classifiers[4])
multi_naive.fit(x_train, y_train)
multi_naive_pred = multi_naive.predict(x_test)

print("Multinominal Naive")
print(f"real classes: {y_test.values}")
print(f"predicted classes: {multi_naive_pred}")
print(f"accuracy: {metrics.accuracy_score(multi_naive_pred, y_test)}")
print(classification_report(y_test, multi_naive_pred, target_names = my_tags))
```

```
Multinominal Naive
real classes: ['i' 'i' 'u']
predicted classes: ['r' 'u' 'u']
accuracy: 0.3333333333333333
             precision    recall  f1-score   support

          u       0.00      0.00      0.00         2
          r       0.00      0.00      0.00         0
          i       0.50      1.00      0.67         1

avg / total       0.17      0.33      0.22         3


/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted sam
ples.
  'precision', 'predicted', average, warn_for)
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are
ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

In [83]:

```
neural_net = pipeline(classifiers[6])
neural_net.fit(x_train, y_train)
neural_net_pred = neural_net.predict(x_test)

print("Neural Net")
print(f"real classes: {y_test.values}")
print(f"predicted classes: {neural_net_pred}")
print(f"accuracy: {metrics.accuracy_score(neural_net_pred, y_test)}")
print(classification_report(y_test, neural_net_pred, target_names = my_tags))
```

```
Neural Net
real classes: ['i' 'i' 'u']
predicted classes: ['r' 'u' 'u']
accuracy: 0.3333333333333333
             precision    recall  f1-score   support

          u       0.00      0.00      0.00         2
          r       0.00      0.00      0.00         0
          i       0.50      1.00      0.67         1

avg / total       0.17      0.33      0.22         3


/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted sam
ples.
  'precision', 'predicted', average, warn_for)
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/metrics/cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are
ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

Classificação com NLTK

In [84]:

```
def pre_process(raw):
    stopwords = nltk.corpus.stopwords.words('portuguese')

    tokens = nltk.word_tokenize(raw.lower())
    filtered = [t for t in tokens if t not in stopwords and t.isalpha() and len(
t) > 1]
    text = nltk.Text(tokens)

    return tokens, filtered, text
```

In [85]:

```
sra_tokens, sra_filtered, sra_text = pre_process(senhora)
diva_tokens, diva_filtered, diva_text = pre_process(diva)
gau_tokens, gau_filtered, gau_text = pre_process(gaucho)
gua_tokens, gua_filtered, gua_text = pre_process(guarani)
ira_tokens, ira_filtered, ira_text = pre_process(iracema)
luci_tokens, luci_filtered, luci_text = pre_process(luciola)
viu_tokens, viu_filtered, viu_text = pre_process(viuvinha)
ubi_tokens, ubi_filtered, ubi_text = pre_process(ubirajara)
sert_tokens, sert_filtered, sert_text = pre_process(sertanejo)
ipe_tokens, ipe_filtered, ipe_text = pre_process(troncodoipe)
```

In [86]:

```
tokens = [sra_filtered, diva_filtered, gau_filtered, gua_filtered, ira_filtered,
 luci_filtered, viu_filtered, ubi_filtered, sert_tokens, ipe_tokens]
```

In [87]:

```
def TFIDF(document):
    word_tfidf = []
    for word in set(collection):
        word_tfidf.append(collection.tf_idf(word,document))
    return word_tfidf
```

In [88]:

```
collection = nltk.text.TextCollection(texts)
```

In [89]:

```
import os
from nltk.corpus.reader.plaintext import PlaintextCorpusReader

corpusdir = 'books/'
corpus = PlaintextCorpusReader(corpusdir, '.*')
```

In [90]:

```
labeled_tokens = list(zip(tokens, labels))
```

In [91]:

```
stopwords = nltk.corpus.stopwords.words('portuguese')
filtered_words = [w.lower() for w in corpus.words() if w not in stopwords and w.
isalpha() and len(w) > 1]

all_words = nltk.FreqDist(filtered_words)
```

In [92]:

```
word_features = list(all_words)[:2000]
```

In [93]:

```
def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

In [94]:

```
documents = [(list(corpus.words(fileid)), category)
             for category in labels
             for fileid in corpus.fileids()]
```

In [95]:

```
featuresets = [(document_features(d), c) for (d,c) in documents]
```

In [96]:

```
train_set, test_set = featuresets[:50], featuresets[50:]
```

In [97]:

```
naivebayes = nltk.NaiveBayesClassifier.train(train_set)
```

In [98]:

```
nltk.classify.accuracy(naivebayes, test_set)
```

Out[98]:

```
0.4
```

In [99]:

```
decisiontree = nltk.DecisionTreeClassifier.train(train_set)
```

In [100]:

```
nltk.classify.accuracy(decisiontree, test_set)
```

Out[100]:

```
0.4
```

Classificação com NLTK e Scikitlearn

In [101]:

```
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC

classifier = nltk.NaiveBayesClassifier.train(train_set)
original_naive = nltk.classify.accuracy(classifier, test_set)
print("Original Naive Bayes Algo accuracy percent:", (original_naive)*100)
classifier.show_most_informative_features(15)
```

```
Original Naive Bayes Algo accuracy percent: 40.0
Most Informative Features
       contains(moleira) = True                r : u      =      1.1
: 1.0
    contains(desapareci) = True                r : u      =      1.1
: 1.0
        contains(ralhei) = True                r : u      =      1.1
: 1.0
          contains(diva) = True                r : u      =      1.1
: 1.0
        contains(homens) = False               r : u      =      1.1
: 1.0
       contains(retirei) = True                r : u      =      1.1
: 1.0
     contains(repuxavam) = True                r : u      =      1.1
: 1.0
           contains(bom) = False               r : u      =      1.1
: 1.0
         contains(baixo) = False               r : u      =      1.1
: 1.0
          contains(mila) = True                r : u      =      1.1
: 1.0
     contains(perfumado) = False               r : u      =      1.1
: 1.0
    contains(desastrado) = True                r : u      =      1.1
: 1.0
     contains(densidade) = True                r : u      =      1.1
: 1.0
         contains(irmão) = False               r : u      =      1.1
: 1.0
  contains(achamalotada) = True                r : u      =      1.1
: 1.0
```

In [102]:

```
MNB_classifier = SklearnClassifier(MultinomialNB())
MNB_classifier.train(train_set)
MNB_accuracy = nltk.classify.accuracy(MNB_classifier, test_set)
print("MNB_classifier accuracy percent:", (MNB_accuracy)*100)
```

```
MNB_classifier accuracy percent: 22.0
```

In [103]:

```
BernoulliNB_classifier = SklearnClassifier(BernoulliNB())
BernoulliNB_classifier.train(train_set)
BernoulliNB_accuracy = nltk.classify.accuracy(BernoulliNB_classifier, test_set)
print("BernoulliNB_classifier accuracy percent:", (BernoulliNB_accuracy)*100)
```

```
BernoulliNB_classifier accuracy percent: 24.0
```

In [104]:

```
SGDClassifier_classifier = SklearnClassifier(SGDClassifier())
SGDClassifier_classifier.train(train_set)
SGDClassifier_accuracy = nltk.classify.accuracy(SGDClassifier_classifier, test_s
et)
print("SGDClassifier_classifier accuracy percent:", (SGDClassifier_accuracy)*100
)
```

```
SGDClassifier_classifier accuracy percent: 40.0
```

```
/home/ellen/anaconda3/lib/python3.7/site-packages/sklearn/linear_mod
el/stochastic_gradient.py:128: FutureWarning: max_iter and tol param
eters have been added in <class 'sklearn.linear_model.stochastic_gra
dient.SGDClassifier'> in 0.19. If both are left unset, they default
to max_iter=5 and tol=None. If tol is not None, max_iter defaults to
max_iter=1000. From 0.21, default max_iter will be 1000, and default
tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

In [105]:

```
SVC_classifier = SklearnClassifier(SVC())
SVC_classifier.train(train_set)
SVC_accuracy = nltk.classify.accuracy(SVC_classifier, test_set)
print("SVC_classifier accuracy percent:", (SVC_accuracy)*100)
```

```
SVC_classifier accuracy percent: 30.0
```

In [106]:

```
LinearSVC_classifier = SklearnClassifier(LinearSVC())
LinearSVC_classifier.train(train_set)
LinearSVC_accuracy = nltk.classify.accuracy(LinearSVC_classifier, test_set)
print("LinearSVC_classifier accuracy percent:", (LinearSVC_accuracy)*100)
```

```
LinearSVC_classifier accuracy percent: 34.0
```

In [107]:

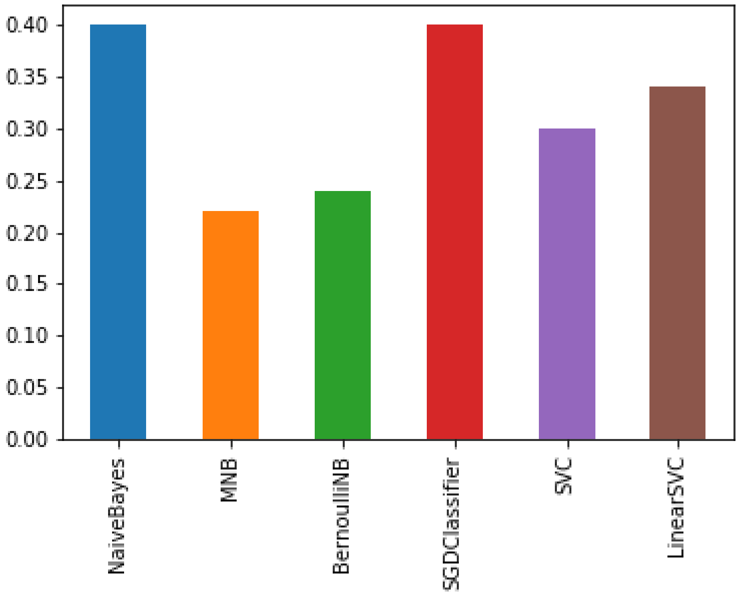
```
d = {
    "NaiveBayes": original_naive,
    "MNB": MNB_accuracy,
    "BernoulliNB": BernoulliNB_accuracy,
    "SGDClassifier": SGDClassifier_accuracy,
    "SVC": SVC_accuracy,
    "LinearSVC": LinearSVC_accuracy
}
```

In [109]:

```
pd.Series(d).plot(kind='bar')
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f70af7352b0>
```



In [ ]:

```

```