

	News	Price Sentiment	Cleaned_News
0	april gold down 20 cents to settle at \$1,116.1...	negative	april gold down cents to settle at
1	gold suffers third straight daily decline	negative	gold suffers third straight daily decline
2	Gold futures edge up after two-session decline	positive	gold futures edge up after twosession decline
3	dent research : is gold's day in the sun comin...	none	dent research is golds day in the sun coming ...
4	Gold snaps three-day rally as Trump, lawmakers...	negative	gold snaps threeday rally as trump lawmakers ...
5	Dec. gold climbs \$9.40, or 0.7%, to settle at ...	positive	dec gold climbs or to settle at
6	gold falls by rs 25 on sluggish demand, global...	negative	gold falls by rs on sluggish demand global cues
7	Gold futures fall for the session, but gain fo...	positive	gold futures fall for the session but gain for...
8	Gold struggles; silver slides, base metals falter	neutral	gold struggles silver slides base metals falter
9	april gold holds slight gain, up \$2.50, or 0.2...	positive	april gold holds slight gain up or at

```
In [18]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)
|
print("x_train:",len(x_train))
print("x_test:",len(x_test))
print("y_train:",len(y_train))
print("y_test:",len(y_test))

x_train: 8456
x_test: 2114
y_train: 8456
y_test: 2114
```

Model building with Logistic Regression

```
In [20]: from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
tvec = TfidfVectorizer()
clf2 = LogisticRegression()
```

```
In [21]: from sklearn.pipeline import Pipeline
model = Pipeline([('vectorizer',tvec),('classifier',clf2)])
model.fit(x_train,y_train)
from sklearn.metrics import confusion_matrix
predictions = model.predict(x_test)
pred_train=model.predict(x_train)
confusion_matrix(predictions, y_test)
```

```
Out[21]: array([[701, 15, 26, 29],
 [ 1, 50, 1, 2],
 [31, 9, 330, 48],
 [36, 15, 34, 786]], dtype=int64)
```

Model building with SVM ¶

```
In [22]: from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
svm=SVC(kernel='linear')
tvec = TfidfVectorizer()
```

```
In [23]: # from sklearn.pipeline import Pipeline
model2 = Pipeline([('vectorizer',tvec),('classifier',svm)])
model2.fit(x_train,y_train)
from sklearn.metrics import confusion_matrix
predictions2 = model.predict(x_test)
pred2_train=model.predict(x_train)
confusion_matrix(predictions2, y_test)
```

```
Out[23]: array([[701, 15, 26, 29],
 [ 1, 50, 1, 2],
 [31, 9, 330, 48],
 [36, 15, 34, 786]], dtype=int64)
```

```
In [24]: #Logistic Regression
from sklearn.metrics import accuracy_score
print("Accuracy_test : ", accuracy_score(predictions, y_test))
print("Accuracy_train : ", accuracy_score(pred_train, y_train))
```

```
Accuracy_test : 0.8831598864711447
Accuracy_train : 0.9331835383159887
```

```
In [25]: #SVM
from sklearn.metrics import accuracy_score
print("Accuracy_test: ", accuracy_score(predictions2, y_test))
print("Accuracy_train : ", accuracy_score(pred2_train, y_train))
```

```
Accuracy_test: 0.8831598864711447
Accuracy_train : 0.9331835383159887
```

```
In [26]: example = ["gold to trade in 28670-29160 range: achievers equities"]
result = model.predict(example)
print(result)
```

```
['neutral']
```

```
In [27]: example = ["gold to trade in 28670-29160 range: achievers equities"]
result = model2.predict(example)
print(result)
```

```
['neutral']
```

```
In [28]: example = ["can investment in gold, sensex & ppfs give the same returns?"]
result = model.predict(example)
print(result)
```

```
['none']
```

```
In [29]: example = ["can investment in gold, sensex & ppfs give the same returns?"]
result = model2.predict(example)
print(result)
```

```
['none']
```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix																																								
LOGISTIC REGRESSION	<pre>#for Logistic Regression from sklearn.metrics import classification_report # assume y_train and pred_train are your true and predicted labels, respectively print(classification_report(y_test, predictions))</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>negative</td><td>0.91</td><td>0.91</td><td>0.91</td><td>769</td></tr><tr><td>neutral</td><td>0.93</td><td>0.56</td><td>0.70</td><td>89</td></tr><tr><td>none</td><td>0.79</td><td>0.84</td><td>0.82</td><td>391</td></tr><tr><td>positive</td><td>0.90</td><td>0.91</td><td>0.91</td><td>865</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.88</td><td>2114</td></tr><tr><td>macro avg</td><td>0.88</td><td>0.81</td><td>0.83</td><td>2114</td></tr><tr><td>weighted avg</td><td>0.88</td><td>0.88</td><td>0.88</td><td>2114</td></tr></tbody></table>		precision	recall	f1-score	support	negative	0.91	0.91	0.91	769	neutral	0.93	0.56	0.70	89	none	0.79	0.84	0.82	391	positive	0.90	0.91	0.91	865	accuracy			0.88	2114	macro avg	0.88	0.81	0.83	2114	weighted avg	0.88	0.88	0.88	2114	88%	<pre>confusion_matrix(predictions, y_test)</pre> <pre>array([[701, 15, 26, 29], [1, 50, 1, 2], [31, 9, 330, 48], [36, 15, 34, 786]], dtype=int64)</pre>
	precision	recall	f1-score	support																																							
negative	0.91	0.91	0.91	769																																							
neutral	0.93	0.56	0.70	89																																							
none	0.79	0.84	0.82	391																																							
positive	0.90	0.91	0.91	865																																							
accuracy			0.88	2114																																							
macro avg	0.88	0.81	0.83	2114																																							
weighted avg	0.88	0.88	0.88	2114																																							

SUPPORT
VECTOR
MACHINE

```
#for SVM
from sklearn.metrics import classification_report
# assume y_train and pred2_train are your true and predicted labels, respectively
print(classification_report(y_test, predictions2))
```

	precision	recall	f1-score	support
negative	0.92	0.90	0.91	769
neutral	0.81	0.64	0.72	89
none	0.79	0.85	0.82	391
positive	0.90	0.90	0.90	865
accuracy			0.88	2114
macro avg	0.86	0.82	0.84	2114
weighted avg	0.88	0.88	0.88	2114

88%

```
confusion_matrix(predictions2, y_test)
```

```
array([[701, 15, 26, 29],
       [ 1, 50, 1, 2],
       [31, 9, 330, 48],
       [36, 15, 34, 786]], dtype=int64)
```