Analysis of different algorithms applied

Algorithms applied

- Logistic Regression
- SVM
- Random Forest
- Multivariate Gaussian
- Decision Trees

Aim

- How each algorithm behaved?
  - Accuracy
  - Precision
  - Recall
  - F1 Score
- Comparison and Conclusion

General

- Data used is ‘Under sampled’ from the original dataset.
- Total 984 items, of which 492 are fraud and rest 492 genuine.
- 28 feature attribute, 1 time, 1 amount and 1 class attribute.
- Time is not playing major role since fraud does not decrease/increase in a systematic manner in the dataset provided, hence it is ignored.
- Amount is normalized for standardization.
- Class=1 represents fraudulent, and Class=0 represents genuine.

Logistic Regression

- Code

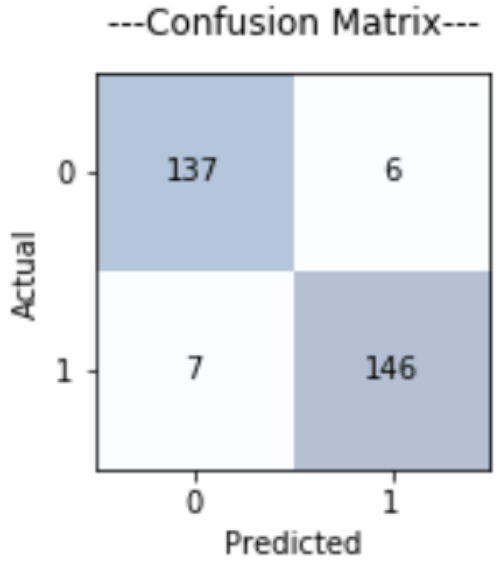
```
from sklearn.linear_model import LogisticRegression
LogReg = LogisticRegression()
LogReg.fit(X_train, y_train.values.ravel())
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
y_pred = LogReg.predict(X_test)
```

Logistic Regression

- Result




```
True Positive= 137 , True Negetive= 146 , False Positive= 7 , False Negetive= 6

The accuracy is 95.6081081081 %
The recall is 95.8041958042 %
The precision is 95.1388888889 %
The F1 Score is 95.4703832753 %
```

SVM

- Code

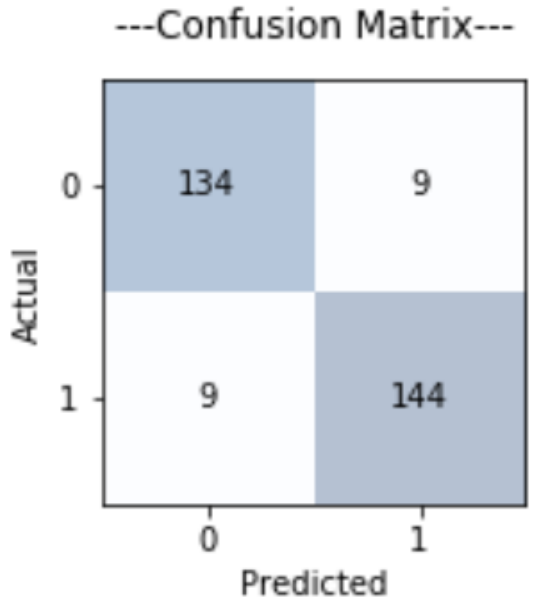
```
from sklearn.svm import SVC
svm_classifier= SVC(C= 1, kernel= 'rbf', random_state= 0)
svm_classifier.fit(X_train, y_train.values.ravel())
```

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=0, shrinking=True,
  tol=0.001, verbose=False)
```

```
y_pred = svm_classifier.predict(X_test)
```

SVM

- Result




```
True Positive= 134 , True Negetive= 144 , False Positive= 9 , False Negetive= 9

The accuracy is 93.9189189189 %
The recall is 93.7062937063 %
The precision is 93.7062937063 %
The F1 Score is 93.7062937063 %
```

Random Forest

- Code

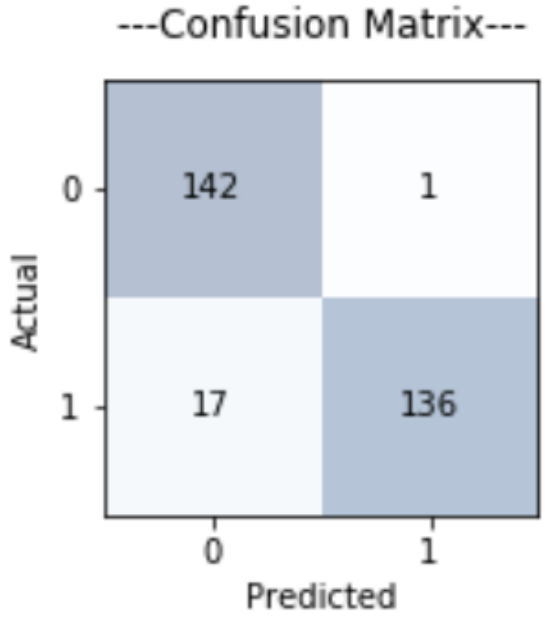
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(random_state=0)
rf.fit(X_train, y_train.values.ravel())
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

```
y_pred = rf.predict(X_test)
```

Random Forest

- Result




```
True Positive= 142 , True Negetive= 136 , False Positive= 17 , False Negetive= 1

The accuracy is 93.9189189189 %
The recall is 99.3006993007 %
The precision is 89.3081761006 %
The F1 Score is 94.0397350993 %
```

Multivariate Gaussian

- Code

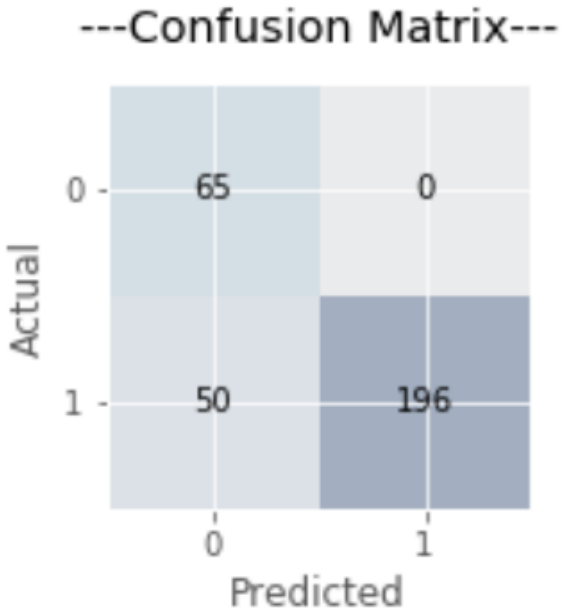
```
from scipy.stats import multivariate_normal
var = multivariate_normal.pdf(X_test_2,X_test_2.mean(), cov_mat)
eps=min(var)
def covariance_matrix(X):
    m=len(X)
    mu = X.mean()
    Sigma=0
    for i in range(m):
        Sigma += np.outer(X[i] - mu, X[i] - mu)
    return Sigma / m
```

```
cov_mat = covariance_matrix(X_train)
cov_mat_inv = np.linalg.pinv(cov_mat)
cov_mat_det = np.linalg.det(cov_mat)
np.matrix(cov_mat).shape
```

```
(28, 28)
```

Multivariate Gaussian

- Result




```
True Positive= 65 , True Negetive= 196 , False Positive= 50 , False Negetive= 0

The accuracy is 83.922829582 %
The recall is 100.0 %
The precision is 56.5217391304 %
The F1 Score is 72.2222222222 %
```

Decision Tree

- Code

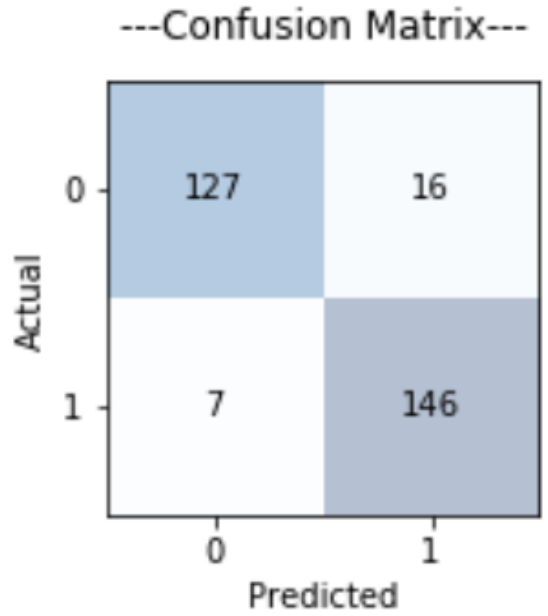
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state = 42)
dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=42,
            splitter='best')
```

```
y_pred = dt.predict(X_test)
```

Decision Tree

- Result




```
True Positive= 127 , True Negetive= 146 , False Positive= 7 , False Negetive= 16

The accuracy is 92.2297297297 %
The recall is 88.8111888112 %
The precision is 94.776119403 %
The F1 Score is 91.6967509025 %
```

Comparison

	Logistic Regression	SVM	Random Forest	MVG	Decision Trees
Accuracy(%)	95.6	93.91	93.91	83.92	92.22
Precision(%)	95.8	93.7	99.3	100	88.81
Recall(%)	95.13	93.7	89.3	56.52	94.77
F1 Score(%)	95.47	93.7	94.03	72.22	91.69

Observation & Conclusion

- Logistic Regression has given least error, followed by SVM.
- Random Forest and Decision Trees have almost behaved similarly ranging from 89%-94%.
- MVG had a lot of false positives but no false negatives. Due to such high false +ves recall and F1 score behaved very poor. However, precision was 100%. This is because recall=tp/(tp+fn) and precision=tp/(tp+fp), since fp=0, precision=100%.