```
#Imports
import pandas as pd
import numpy as np
import itertools
import time
import seaborn as sn
import matplotlib.pyplot as plt

from sklearn import preprocessing, neighbors, decomposition
#from sklearn import metrics
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.preprocessing import scale
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_predict
%matplotlib inline
```

```
# Load Data
caravan = pd.read_csv('Caravan.csv', index_col=0)
```

```
caravan.head()
```

| | MOSTYPE | MAANTHUI | MGEMOMV | MGEMLEEF | MOSHOOFD | MGODRK | MGODPR | MGODOV | MGODGE |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 33 | 1 | 3 | 2 | 8 | 0 | 5 | 1 | 3 |
| **2** | 37 | 1 | 2 | 2 | 8 | 1 | 4 | 1 | 4 |
| **3** | 37 | 1 | 2 | 2 | 8 | 0 | 4 | 2 | 4 |
| **4** | 9 | 1 | 3 | 3 | 3 | 2 | 3 | 2 | 4 |
| **5** | 40 | 1 | 4 | 2 | 10 | 1 | 4 | 1 | 4 |

5 rows × 86 columns

1. Understanding the Dataset: Given the provided datasets (as CSV files), load them and answer the following questions.

(a) What is the dimension of the datasets? 5822 observations, 85 predictors

```
caravan.shape
```

```
(5822, 86)
```

(b) How many predictors measure demographic characteristics? Using the fact that demographic characteristics begin with M we find that there are 43 of them.

```
predictors_demo = [col for col in caravan.columns if col[0] == 'M']
len(predictors_demo)
```

```
43
```

(c) What is the percentage of people who purchased caravan insurance?. In this data set, only 5.977% of people purchased caravan insurance.

```
count_purchased = len(
    [purchased for purchased in caravan["Purchase"] if purchased.strip() == "Yes"])
percent = (count_purchased * 100)/len(caravan["Purchase"])
percent
```

```
5.977327378907592
```

2. Data preprocessing: Standardize the data matrix X so that all variables are given a mean of zero and a standard deviation of one. In standardizing the datasets, exclude the response variable.

```
y = caravan.Purchase
X = caravan.drop('Purchase', axis=1).astype('float64')
X_scaled = preprocessing.scale(X)
X_scaled.mean(), X_scaled.std()
```

```
(-7.064219411036621e-18, 0.9999999999999999)
```

3. Split the datasets into a test set, containing the first 1,000 observations, and a training set, containing the remaining observations.

```
X_train = X_scaled[1000:, :]
y_train = y[1000:]

X_test = X_scaled[:1000, :]
y_test = y[:1000]
```

(a) How many observations are in each set? The testing data has 1000 observations The training data has 4822 observations

```
X_train.shape
```

```
(4822, 85)
```

```
X_test.shape
```

```
(1000, 85)
```

(b) How many customers purchased insurance in each set? 59 people from test dataset 289 people from the train dataset

```
y_train.value_counts()
```

```
No     4533
Yes     289
Name: Purchase, dtype: int64
```

```
y_test.value_counts()
```

```
No     941
Yes     59
Name: Purchase, dtype: int64
```

4. Binary Classifier: KNN and SGD classifiers

(a) Apply the K-Nearest Neighbors (KNN) classifier to the caravan dataset. Choose the values K = 1, 3 and 5 and for each K, compute the precision and recall. Please comment on the precision. Hint: Details of the KNN classifier can be found here: [https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)

```
knn = neighbors.KNeighborsClassifier(n_neighbors=1)
pred = knn.fit(X_train, y_train).predict(X_test)
# Show confusion matrix
c_matrix_1 = confusion_matrix(y_test, pred)
c_matrix_1
```

```
array([[873,  68],
       [ 50,   9]])
```

```
#Precision for K = 1
precision = c_matrix_1[1, 1]/(c_matrix_1[0, 1]+c_matrix_1[1, 1])
precision
```

```
0.11688311688311688
```

```
#Recall for K = 1
recall = c_matrix_1[1, 1]/(c_matrix_1[1, 0]+c_matrix_1[1, 1])
recall
```

# K = 3

```
knn = neighbors.KNeighborsClassifier(n_neighbors=3)
pred = knn.fit(X_train, y_train).predict(X_test)
c_matrix_3 = confusion_matrix(y_test, pred)
c_matrix_3
```

```
array([[921,  20],
       [ 54,   5]])
```

```
#Precision for K = 3
precision = c_matrix_3[1, 1]/(c_matrix_3[0, 1]+c_matrix_3[1, 1])
precision
```

```
0.2
```

```
#Recall for K = 3
recall = c_matrix_3[1, 1]/(c_matrix_3[1, 0]+c_matrix_3[1, 1])
recall
```

```
0.0847457627118644
```

# K = 5

```
knn = neighbors.KNeighborsClassifier(n_neighbors=5)
pred = knn.fit(X_train, y_train).predict(X_test)
c_matrix_5 = confusion_matrix(y_test, pred)
c_matrix_5
```

```
array([[930,  11],
       [ 55,   4]])
```

```
#Precision for K = 5
precision = c_matrix_5[1, 1]/(c_matrix_5[0, 1]+c_matrix_5[1, 1])
precision
```

```
0.26666666666666666
```

```
#Recall for K = 5
recall = c_matrix_5[1, 1]/(c_matrix_5[1, 0]+c_matrix_5[1, 1])
recall
```

```
0.06779661016949153
```

precision: k=1 : 11.7% k=3 : 20% k=5 : 26.6% As the number of neighbors (k) increases,precision increases as well.

# SGD

(b) Next apply the Stochastic Gradient Descent (SGD) classifier on the caravan dataset. Compute the precision and recall. Set random seed to 42.

```
#convert the "purchased" field  to boolean
y_train_purchased = (y_train == "Yes")
y_test_purchased = (y_test == "Yes")
```

```
model = SGDClassifier(loss="hinge", alpha=0.01, max_iter=200,random_state=42)
model.fit(X_train, y_train)
# Predicting the results
model.predict(X_test[0:1])
```

```
array(['No'], dtype='<U3')
```

```
y_train_pred = cross_val_predict(model, X_train, y_train_purchased)
```

```
from sklearn.metrics import precision_score, recall_score
precision_score(y_train_purchased, y_train_pred)
```

```
0.21052631578947367
```

```
recall_score(y_train_purchased, y_train_pred)
```

```
0.01384083044982699
```

```
# Confusion matrix
print("Confusion Matrix")
matrix = confusion_matrix(y_train_purchased, y_train_pred)
print(matrix)
# Classification Report
print("\nClassification Report")
report = classification_report(y_train_purchased, y_train_pred)
print(report)
# Accuracy of the model
accuracy = accuracy_score(y_train_purchased, y_train_pred)
print('SGD Classifier Accuracy of the model: {:.2f}%'.format(accuracy*100))
```

```
Confusion Matrix
[[4518   15]
 [ 285    4]]

Classification Report
              precision    recall  f1-score   support

       False       0.94      1.00      0.97      4533
        True       0.21      0.01      0.03       289

    accuracy                           0.94      4822
   macro avg       0.58      0.51      0.50      4822
weighted avg       0.90      0.94      0.91      4822

SGD Classifier Accuracy of the model: 93.78%
```

(c) Which classifier finds real patterns in the caravan dataset?

KNN seems to perform better, with a higher precission when k=5