Application of Ml in Industries

Experiment 3

Dhruv Singhal || 500075346 || R177219074 || AIML || B3 || Sem 5

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df = pd.read_csv("winequalityN.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	

PreProcess

In [4]:

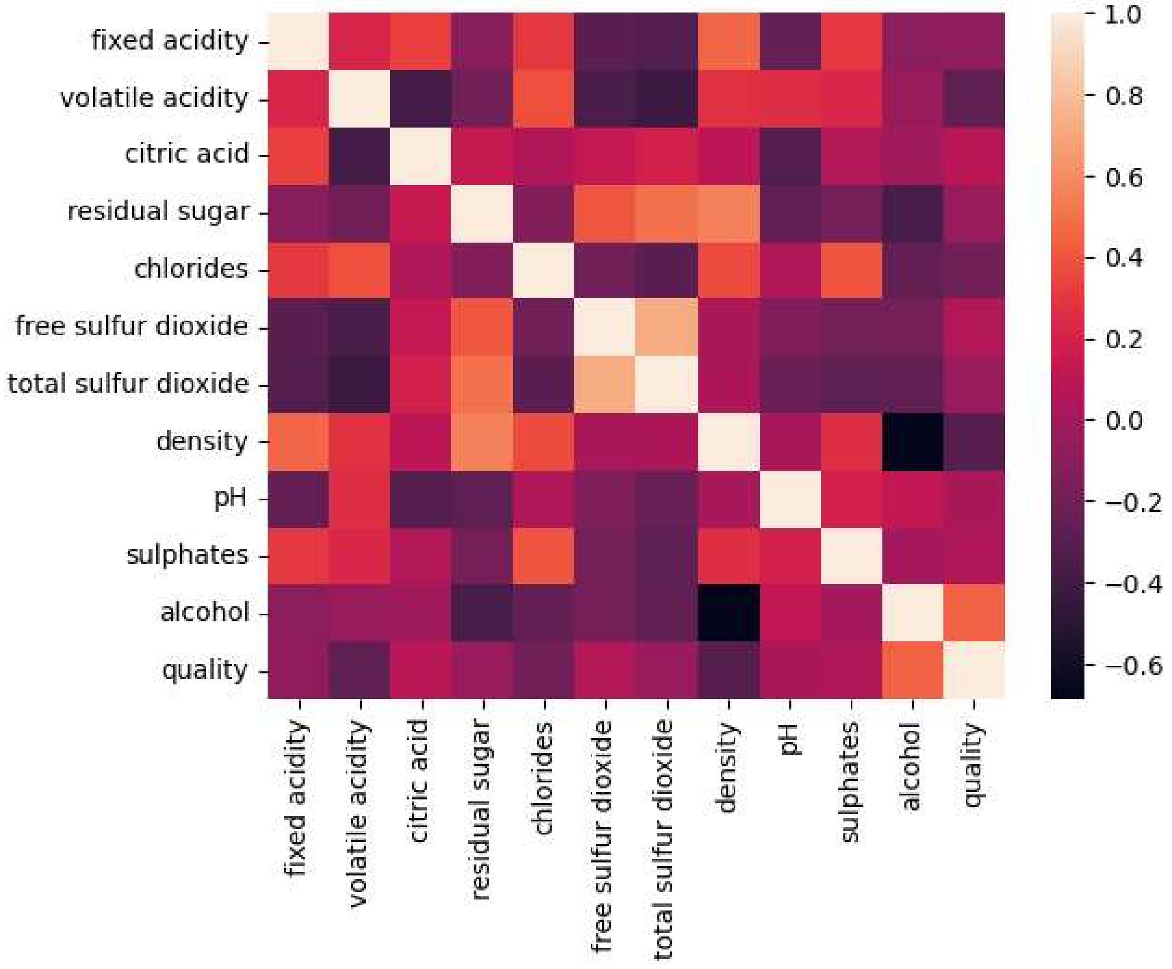
```
df.shape
```

Out[4]:

```
(6497, 13)
```

In [7]:

```
sns.heatmap(df.corr())
plt.show()
```



In [8]:

```
df.isna().sum()
```

Out[8]:

```
type                     0
fixed acidity           10
volatile acidity         8
citric acid              3
residual sugar           2
chlorides                2
free sulfur dioxide      0
total sulfur dioxide     0
density                  0
pH                       9
sulphates                4
alcohol                  0
quality                  0
dtype: int64
```

In [9]:

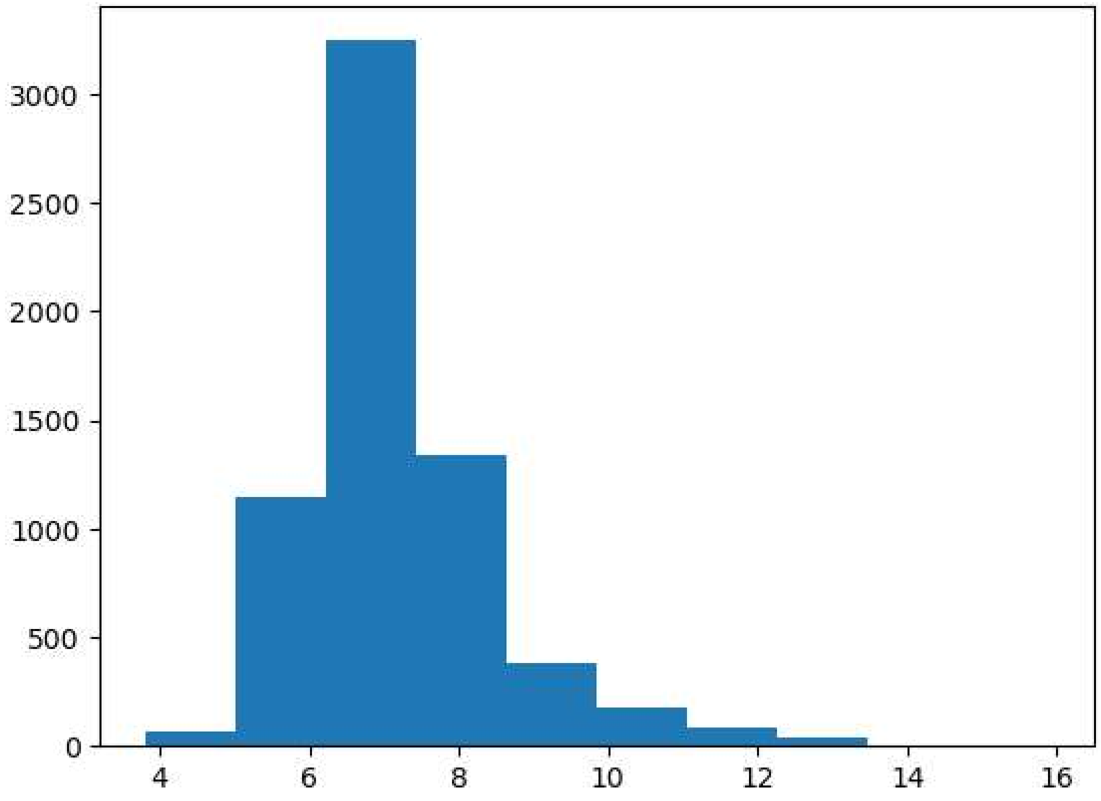
```
df.columns
```

Out[9]:

```
Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',
       'residual sugar', 'chlorides', 'free sulfur dioxide',
       'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
       'quality'],
      dtype='object')
```

In [10]:

```
plt.hist(df["fixed acidity"])
plt.show()
```

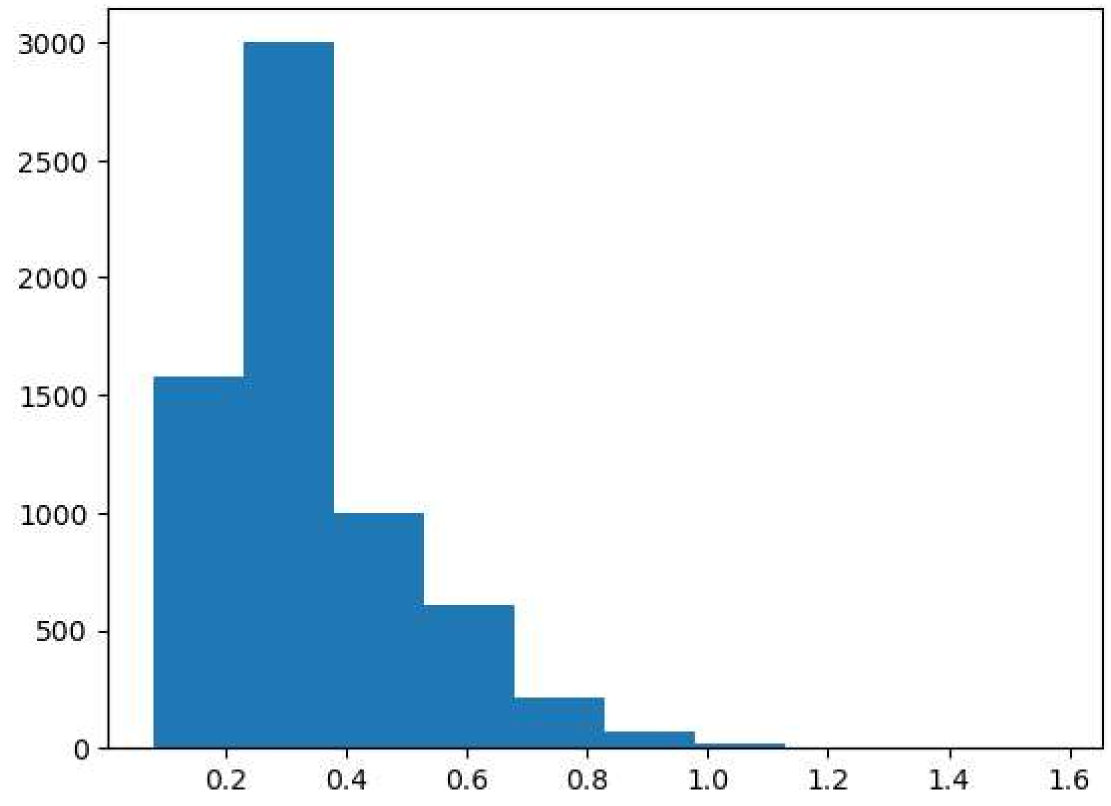


In [11]:

```
mean = df["fixed acidity"].mean()
df["fixed acidity"].fillna(mean,inplace=True)
```

In [12]:

```
plt.hist(df["volatile acidity"])
plt.show()
```

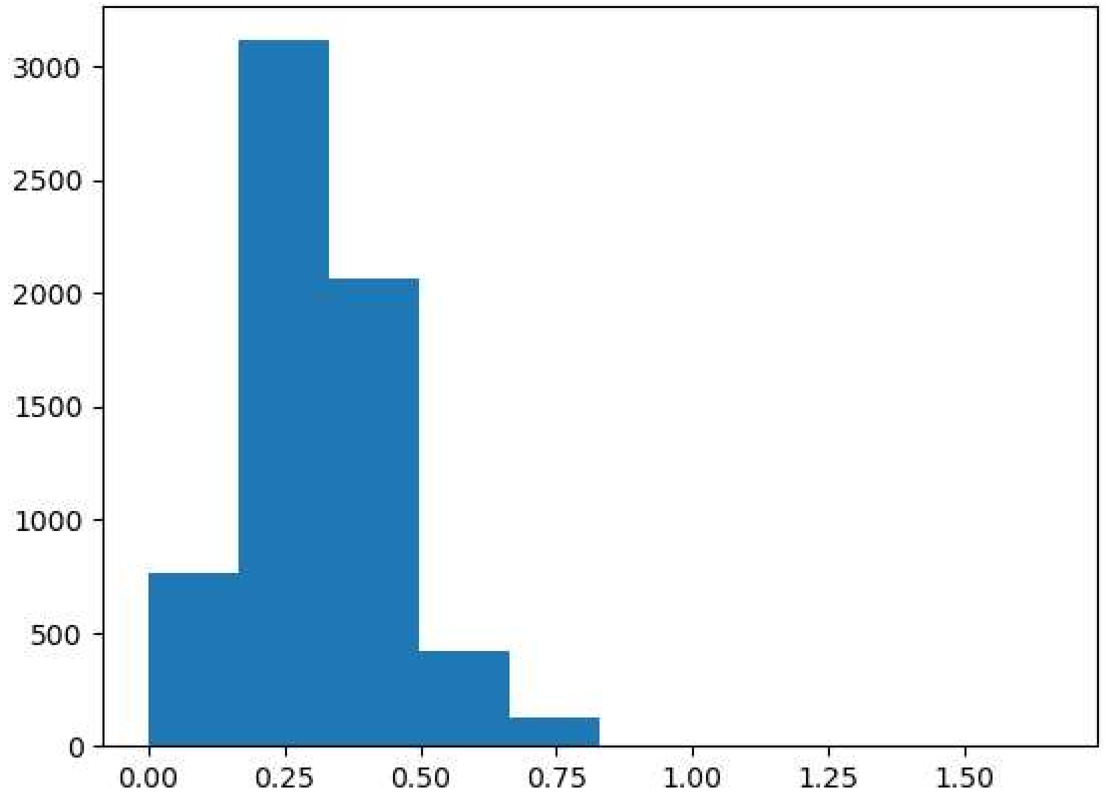


In [13]:

```
mean2 = df["volatile acidity"].mean()
df["volatile acidity"].fillna(mean2,inplace=True)
```

In [14]:

```
plt.hist(df["citric acid"])
plt.show()
```



In [15]:

```
mean3 = df["citric acid"].mean()
df["citric acid"].fillna(mean3,inplace=True)
```

In [16]:

```
mean4 = df["residual sugar"].mean()
df["residual sugar"].fillna(mean4,inplace=True)
```

In [17]:

```
mean4 = df["chlorides"].mean()
df["chlorides"].fillna(mean4,inplace=True)
```

In [18]:

```
mean5 = df["pH"].mean()
df["pH"].fillna(mean5,inplace=True)
```

In [19]:

```
mean6 = df["sulphates"].mean()
df["sulphates"].fillna(mean6,inplace=True)
```

In [20]:

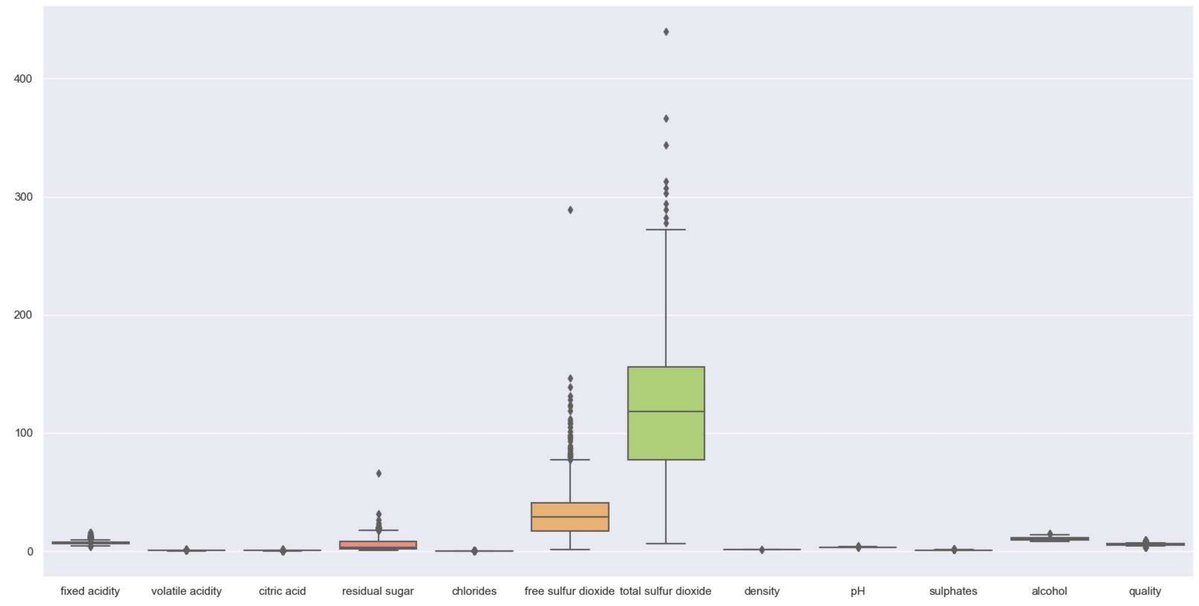
```
df.isna().sum()
```

Out[20]:

```
type                    0
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [21]:

```
sns.set()
plt.figure(figsize=(20,10))
sns.boxplot(data=df,palette="Set3")
plt.show()
```



Outlier Removal

In [22]:

```
lower_limit = df["free sulfur dioxide"].mean() - 3*df["free sulfur dioxide"].std()
upper_limit = df["free sulfur dioxide"].mean() + 3*df["free sulfur dioxide"].std()
```

In [23]:

```
df2 = df[(df["free sulfur dioxide"] > lower_limit) & (df["free sulfur dioxide"] < upper_
```

In [24]:

```
df2.shape
```

Out[24]:

```
(6461, 13)
```

In [25]:

```
lower_limit = df2['residual sugar'].mean() - 3*df2['residual sugar'].std()
upper_limit = df2['residual sugar'].mean() + 3*df2['residual sugar'].std()
```

In [26]:

```
df3 = df2[(df2['residual sugar'] > lower_limit) & (df2['residual sugar'] < upper_limit)]
df3.shape
```

Out[26]:

```
(6435, 13)
```

In [27]:

```
df3.drop("type",axis=1,inplace=True)
```

In [28]:

```
df3.head()
```

Out[28]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1

In [29]:

```
df3["quality"].unique()
```

Out[29]:

```
array([6, 5, 7, 8, 4, 3, 9], dtype=int64)
```

Encoding

In [30]:

```
quaity_mapping = { 3 : "Low",4 : "Low",5: "Medium",6 : "Medium",7: "Medium",8 : "High",9
df3["quality"] =  df3["quality"].map(quaity_mapping)
```

In [31]:

```
df3.quality.value_counts()
```

Out[31]:

```
Medium    6001
Low        240
High       194
Name: quality, dtype: int64
```

In [32]:

```
mapping_quality = {"Low" : 0,"Medium": 1,"High" : 2}
df3["quality"] =  df3["quality"].map(mapping_quality)
```

Feature Importance

In [33]:

```
x = df3.drop("quality",axis=True)
y = df3["quality"]
```

In [34]:

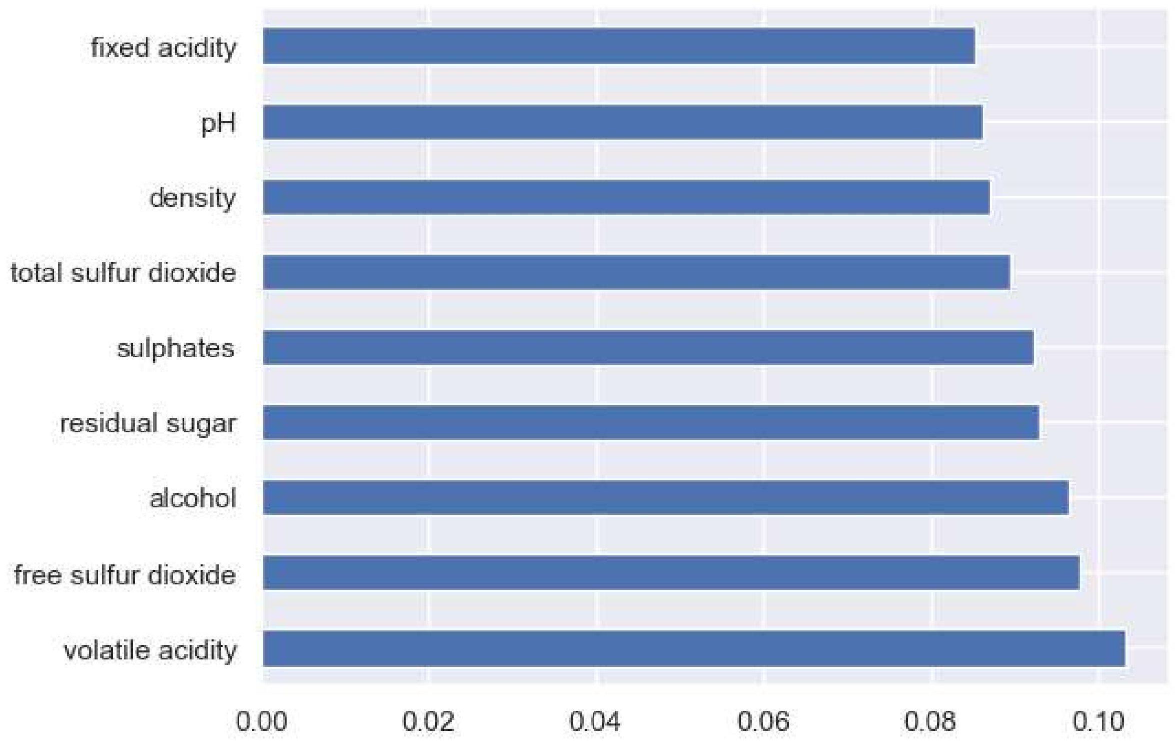
```
model = ExtraTreesClassifier()
model.fit(x,y)
```

Out[34]:

```
ExtraTreesClassifier()
```

In [35]:

```
f_i = pd.Series(model.feature_importances_,index =x.columns)
f_i.nlargest(9).plot(kind="barh")
plt.show()
```



Train test Split

In [36]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=
```

Model

In [37]:

```
model_params  = {
    "svm" : {
        "model":SVC(gamma="auto"),
        "params":{
            'C' : [1,10,20],
            'kernel':["rbf"]
        }
    },

    "decision_tree":{
        "model": DecisionTreeClassifier(),
        "params":{
            'criterion':["entropy","gini"],
            "max_depth":[5,8,9]
        }
    },

    "random_forest":{
        "model": RandomForestClassifier(),
        "params":{
            "n_estimators":[1,5,10],
            "max_depth":[5,8,9]
        }
    },
    "naive_bayes":{
        "model": GaussianNB(),
        "params":{}
    },

    'logistic_regression' : {
        'model' : LogisticRegression(solver='liblinear',multi_class = 'auto'),
        'params': {
            "C" : [1,5,10]
        }
    }

}
```

In [53]:

```
score=[]
for model_name,mp in model_params.items():
    clf = GridSearchCV(mp["model"],mp["params"],cv=8,return_train_score=False)
    clf.fit(x,y)
    print(clf.best_score_)
    score.append({
        "Model" : model_name,
        "Score": clf.best_score_
    })
```

```
0.9314688204938042
0.918409698402398
0.931934659003121
0.7736760684156856
0.9324010769135689
```

In [62]:

```
df5 = pd.DataFrame(score)
```

In [ ]:

```

```

In [63]:

```
df5
```

Out[63]:

	Model	Score
0	svm	0.931469
1	decision_tree	0.918410
2	random_forest	0.931935
3	naive_bayes	0.773676
4	logistic_regression	0.932401

In [ ]:

```

```

In [ ]:

```

```