



Machine learning Foundation Project Report (INT247)

Student Name: Mohit Chaturvedi

Registration no.: 11807252

Course name: Machine learning Foundation (INT247)

Section: K0M32

Roll no. –B69

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab

ML Project (UCI-Nursery dataset)

Task to be done:- (Objective of our project)

1. Preparing data (Feature Identification)
2. Choosing a model (Comparative study min. 3 model)
3. Training (Reiterating for optimization)
4. Evaluation (Highest accuracy with confusion matrix)
5. Hyper parameter tuning
6. Prediction on unknown data
7. Report/presentation

Our Objective:-

Build an appropriate machine learning model by which Prediction uses Machine Learning classification algorithms to categorize whether the candidate is priority, recommended or not recommended to be admitted. The database contains 12960 instances and 8 attributes.

Dataset description:-

1. Title: Nursery Database

3. Past Usage:

The hierarchical decision model, from which this dataset is derived, was first presented in

M. Olave, V. Rajkovic, M. Bohanec: An application for admission in public school systems. In (I. Th. M. Snellen and W. B. H. J. van de Donk and J.-P. Baquias, editors) Expert Systems in Public Administration, pages 145-160. Elsevier Science Publishers (North Holland)}, 1989.

Within machine-learning, this dataset was used for the evaluation of HINT (Hierarchy INduction Tool), which was proved to be able to completely reconstruct the original hierarchical model. This, together with a comparison with C4.5, is presented in

B. Zupan, M. Bohanec, I. Bratko, J. Demsar: Machine learning by function decomposition. ICML-97, Nashville, TN. 1997 (to appear)

Relevant Information Paragraph:

Nursery Database was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It was used during several years in 1980's when there was excessive enrollment to these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation. The final decision depended on three subproblems: occupation of parents and child's nursery, family structure and financial standing, and social and health picture of the family

The hierarchical model ranks nursery-school applications according to the following concept structure:

NURSERY	Evaluation of applications for nursery schools
. EMPLOY	Employment of parents and child's nursery

- . . parents Parents' occupation
- . . has_nurs Child's nursery
- . STRUCT_FINAN Family structure and financial standings
- . . STRUCTURE Family structure
- . . . form Form of the family
- . . . children Number of children
- . . housing Housing conditions
- . . finance Financial standing of the family
- . SOC_HEALTH Social and health picture of the family
- . . social Social conditions
- . . health Health conditions

Number of Instances: 12960

Number of Attributes: 8

Missing Attribute Values: none

Class Distribution (number of instances per class)

class	N	N[%]

not_recom	4320	(33.333 %)
recommend	2	(0.015 %)

very_recom 328 (2.531 %)

priority 4266 (32.917 %)

spec_prior 4044 (31.204 %)

Attributes:-

parents: usual, pretentious, great_pret.

has_nurs: proper, less_proper, improper, critical, very_crit.

form: complete, completed, incomplete, foster.

children: 1, 2, 3, more.

housing: convenient, less_conv, critical.

finance: convenient, inconv.

social: nonprob, slightly_prob, problematic.

health: recommended, priority, not_recom.

Things which I have performed on my Project:-

First we will convert into comma separated file and add attributes as given following column. And then load our dataset into .py file in jupyter using pandas library data frames .also perform some basic tasks like dataset.head() which gives us top rows of our dataset.

And use dataset.describe() which describes the dataset statical views and shows whether the null values are there or not .

Choosing a model:-

To prepare data basically we use Label Encoder and get_dummies() for converting categorical data into numeric data.

And for the feature selection we use ExtraTree classifier and chi-square using scikit-learn tool.

And using scikit learn library we Split the dataset into the Training set and Test set.

So Now we will be Fitting Random Forest Classification to our Training set.

I choose random forest classification because Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

That's why I choose random forest classification algorithm.

We simply create two basic functions one for Fitting Random Forest Classification to our Training set and Predicting the Test set results using scikit-learn And second one for calculating the accuracy of the model using Predicted the Test set results.

Code for random forest classification:-

```
def random_forest(X_train, X_test, y_train, y_test):  
    from sklearn.ensemble import RandomForestClassifier  
  
    classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy',  
                                     random_state = 0)  
  
    classifier.fit(X_train, y_train)  
    y_pred = classifier.predict(X_test)  
  
    print("Random Forest:")  
    accuracy(y_test,y_pred)  
  
def accuracy(y_test,y_pred):
```

```
from sklearn.metrics import accuracy_score
```

```
print("Accuracy=",accuracy_score(y_test, y_pred)*100)
```

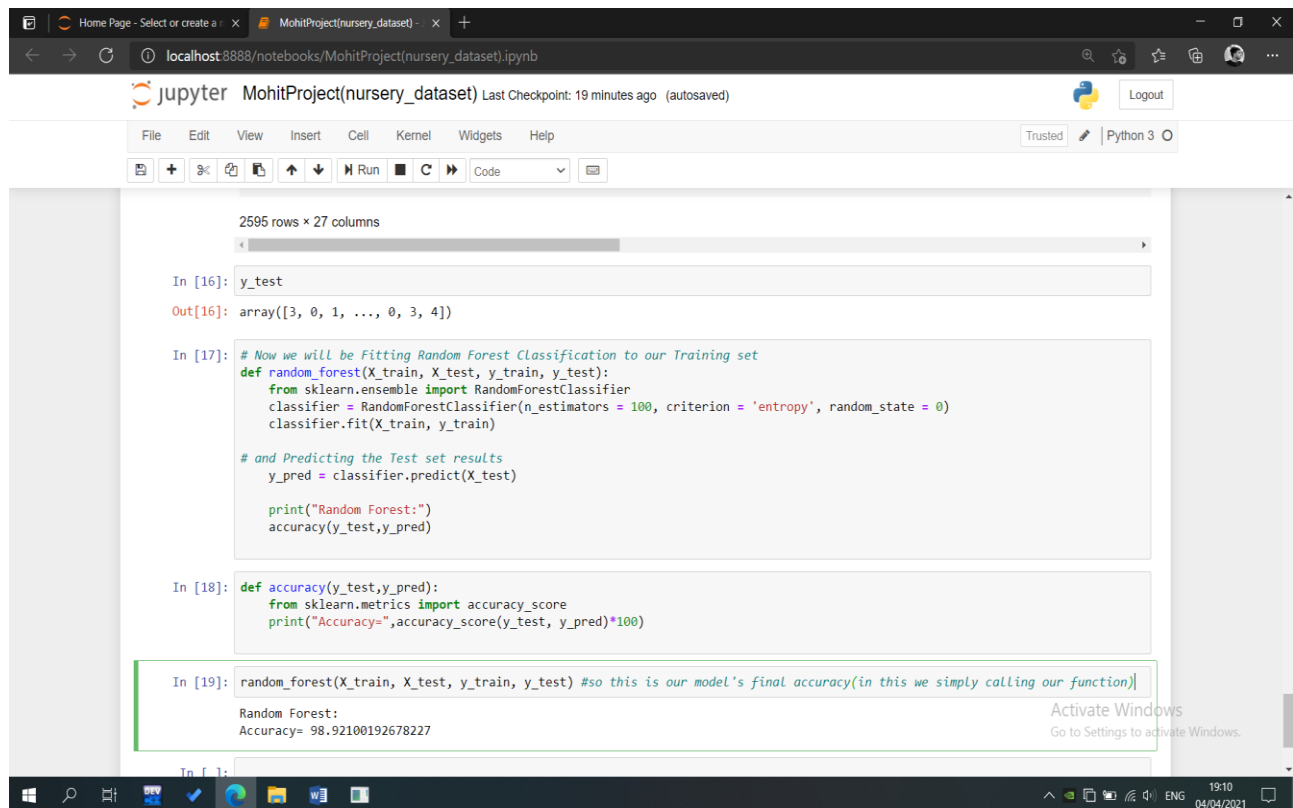
After writing functions we simply call our function:-

```
random_forest(X_train, X_test, y_train, y_test)
```

Output:-

Random Forest:

Accuracy= 98.92100192678227



The screenshot shows a Jupyter Notebook window titled 'MohitProject(nursery_dataset)'. The notebook contains three code cells. The first cell defines a function `random_forest` that fits a `RandomForestClassifier` on training data and predicts on test data. The second cell defines an `accuracy` function that calculates the accuracy score. The third cell calls `random_forest(X_train, X_test, y_train, y_test)` and prints the result. The output of the third cell is 'Random Forest: Accuracy= 98.92100192678227'. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom showing the current kernel is 'Python 3'.

```
2595 rows x 27 columns
```

```
In [16]: y_test
Out[16]: array([3, 0, 1, ..., 0, 3, 4])

In [17]: # Now we will be Fitting Random Forest Classification to our Training set
def random_forest(X_train, X_test, y_train, y_test):
    from sklearn.ensemble import RandomForestClassifier
    classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state = 0)
    classifier.fit(X_train, y_train)

    # and Predicting the Test set results
    y_pred = classifier.predict(X_test)

    print("Random Forest:")
    accuracy(y_test, y_pred)

In [18]: def accuracy(y_test, y_pred):
    from sklearn.metrics import accuracy_score
    print("Accuracy=", accuracy_score(y_test, y_pred)*100)

In [19]: random_forest(X_train, X_test, y_train, y_test) #so this is our model's final accuracy(in this we simply calling our function)

Random Forest:
Accuracy= 98.92100192678227
```

Thank you😊

Mohit Chaturvedi