KAKATIYA GOVERNMENT COLLEGE
HANAMKONDA

Workshop
on

“Hands-on Workshop on Python Libraries for Machine Learning“

(08-09-2022 to 21-09-22)

Organised by

Smt.K.Sravana Kumari
Sri.V.Ramesh

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
2022-23

KAKATIYA GOVERNMET COLLEGE-HANAMAKONDA

Department of Computer Science and Applications

C I R C U L A R

Date:02-09-2022

Department of Computer Science and Applications is organizing ten days workshop on “Hands-on Workshop on Python Libraries for Machine Learning“ from 08-09-2022 to 21-09-2022 for B.COM CA V Sem Students. All the Third year students of B.COM CA are informed to take an active participation to make this activity successful.

PRINCIPAL
KAKATIYA GOVT. COLLEGE
Hanamkonda.

Hands-on Workshop on Python Libraries for Machine Learning

About

With the advances in the cognitive computing domain, it is now possible to develop advanced data analysis tools that can aid specialists in decision-making. Machine learning and deep learning form the bases on which such complex systems are developed. In view of the same, the workshop aims to develop the foundations of using ML-python libraries for interested students.

Agenda

1. Exploratory Data Analysis
2. Data Visualization tools in python
3. Different ML models in Python (No theory)
4. Selecting the best model

Organisers:

Smt.K.Sravana Kumari Sri.V.Ramesh

Objectives

We'll cover the core Python language and the standard library in detail. This course will cover various skills including text manipulation, modular programming, working with and retrieving data, interacting with files on your computer, and using some of the more popular third-party libraries (and getting them installed when and where we need them). The goal is to get participants up and running with Python in as short a time as possible.

Activities

Students will learn the basics of writing and running Python scripts. We will cover topics for people completely new to programming along with comparisons and contrasts to other programming languages. Everything from "OMG white space?!?!" to ways to manipulate the language into a very terse format (also why you might not want to do that) to cool tricks we can do with the simplest, most basic Python data-types.

The Python standard library likely has everything you need, but we won't stop there. We'll make use of some of the more popular third-party libraries, which will also let us make use of the tool pip for grabbing libraries from the Python Package Index (PyPI).

Task1: Binary Prediction of Smoker Status

 # This Python 3 environment comes with many helpful analytics libraries installed
 # It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python

```
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/playground-series-s3e24/sample_submission.csv
/kaggle/input/playground-series-s3e24/train.csv
/kaggle/input/playground-series-s3e24/test.csv
```

In [2]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score, auc
```

In [3]:

```
# Load the train and test data
train_data = pd.read_csv("/kaggle/input/playground-series-s3e24/train.csv")
test_data = pd.read_csv("/kaggle/input/playground-series-s3e24/test.csv")
```

In [4]:

```
# Define the target column
target_column = 'smoking'

# Exclude 'id' column from train data
train_data = train_data.drop(columns=['id'])

# Separate features and target variable
X = train_data.drop(columns=[target_column])
y = train_data[target_column]

# Split the train data into train and validation sets
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:

```
# Initialize the Logistic Regression model
model = LogisticRegression(max_iter=5000)  # Increase max_iter value
```

```
# Train the model on the train data
model.fit(X_train, y_train)
```

Out[5]:

☑ LogisticRegression

LogisticRegression(max_iter=5000)

In [6]:

```
# Predict probabilities on the validation set
y_pred_prob = model.predict_proba(X_valid)[:, 1]

# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_valid, y_pred_prob)
roc_auc = auc(fpr, tpr)

print(f'ROC AUC Score: {roc_auc}')
```

ROC AUC Score: 0.831987247786051

In [7]:

```
# Now, let's make predictions on the test data
# Exclude 'id' column from test data
test_predictions = model.predict_proba(test_data.drop(columns=['id']))[:, 1]

# Create a submission DataFrame
submission = pd.DataFrame({'id': test_data['id'], 'smoking': test_predictions})

# Save the submission to a CSV file
submission.to_csv('submission.csv', index=False)
```

Task2: Sentiment Analysis of Restaurant Reviews

The purpose of this analysis is to build a prediction model to predict whether a review on the restaurant is positive or negative. To do so, we will work on Restaurant Review dataset, we will load it into predicitve algorithms Multinomial Naive Bayes, Bernoulli Naive Bayes and Logistic Regression. In the end, we hope to find a "best" model for predicting the review's sentiment.

Dataset: Restaurant_Reviews.tsv is a dataset from Kaggle datasets which consists of 1000 reviews on a restaurant.

To build a model to predict if review is positive or negative, following steps are performed.

- Importing Dataset
- Preprocessing Dataset
- Vectorization
- Training and Classification
- Analysis Conclusion

Importing Dataset

Importing the Restaurant Review dataset using pandas library.

In [1]:
```
# Importing the libraries
import numpy as np
import pandas as pd
```
In [2]:
```
# Importing the dataset
dataset = pd.read_csv('../input/Restaurant_Reviews.tsv', delimiter = '\t', quoting = 3)
```
Preprocessing Dataset

Each review undergoes through a preprocessing step, where all the vague information is removed.

- Removing the Stopwords, numeric and speacial charecters.
- Normalizing each review using the approach of stemming.

In [3]:
```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)
```
Vectorization

From the cleaned dataset, potential features are extracted and are converted to numerical format. The vectorization techniques are used to convert textual data to numerical format. Using vectorization, a matrix is created where each column represents a feature and each row represents an individual review.

In [4]:
```
# Creating the Bag of Words model using CountVectorizer

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values
```
Training and Classification

Further the data is splitted into training and testing set using Cross Validation technique. This data is used as input to classification algorithm.

Classification Algorithms:

Algorithms like Decision tree, Support Vector Machine, Logistic Regression, Naive Bayes were implemented and on comparing the evaluation metrics two of the algorithms gave better predictions than others.

- Multinomial Naive Bayes
- Bernoulli Naive Bayes
- Logistic Regression

In [5]:

```
# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Multinomial NB

In [6]:

```
# Multinomial NB

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB(alpha=0.1)
classifier.fit(X_train, y_train)

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

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix:\n",cm)

# Accuracy, Precision and Recall
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
score1 = accuracy_score(y_test,y_pred)
score2 = precision_score(y_test,y_pred)
score3= recall_score(y_test,y_pred)
print("\n")
print("Accuracy is ",round(score1*100,2),"%")
print("Precision is ",round(score2,2))
print("Recall is ",round(score3,2))
```

```
Confusion Matrix:
 [[119  33]
 [ 34 114]]


Accuracy is  77.67 %
Precision is  0.78
Recall is  0.77
```

Bernoulli NB

In [7]:

```
# Bernoulli NB

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB(alpha=0.8)
classifier.fit(X_train, y_train)

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

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix:\n",cm)

# Accuracy, Precision and Recall
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
score1 = accuracy_score(y_test,y_pred)
score2 = precision_score(y_test,y_pred)
score3= recall_score(y_test,y_pred)
print("\n")
print("Accuracy is ",round(score1*100,2),"%")
print("Precision is ",round(score2,2))
print("Recall is ",round(score3,2))
```

```
Confusion Matrix:
 [[115  37]
 [ 32 116]]


Accuracy is  77.0 %
Precision is  0.76
Recall is  0.78
```

Logistic Regression

In [8]:

```
# Logistic Regression

# Fitting Logistic Regression to the Training set
from sklearn import linear_model
classifier = linear_model.LogisticRegression(C=1.5)
classifier.fit(X_train, y_train)

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

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix:\n",cm)

# Accuracy, Precision and Recall
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
score1 = accuracy_score(y_test,y_pred)
score2 = precision_score(y_test,y_pred)
score3= recall_score(y_test,y_pred)
print("\n")
print("Accuracy is ",round(score1*100,2),"%")
print("Precision is ",round(score2,2))
print("Recall is ",round(score3,2))
```

```
Confusion Matrix:
 [[125  27]
 [ 43 105]]


Accuracy is  76.67 %
Precision is  0.8
Recall is  0.71
```

Analysis and Conclusion

In this study, an attempt has been made to classify sentiment analysis for restaurant reviews using machine learning techniques. Two algorithms namely Multinomial Naive Bayes and Bernoulli Naive Bayes are implemented.

Evaluation metrics used here are accuracy, precision and recall.

Using Multinomial Naive Bayes,

- Accuracy of prediction is 77.67%.
- Precision of prediction is 0.78.
- Recall of prediction is 0.77.

Using Bernoulli Naive Bayes,

- Accuracy of prediction is 77.0%.
- Precision of prediction is 0.76.
- Recall of prediction is 0.78.

Using Logistic Regression,

- Accuracy of prediction is 76.67%.
- Precision of prediction is 0.8.
- Recall of prediction is 0.71.

From the above results, Multinomial Naive Bayes is slightly better method compared to Bernoulli Naive Bayes and Logistic Regression, with 77.67% accuracy which means the model built for the prediction of sentiment of the restaurant review gives 77.

Task3: MANIPULATING DATA FRAMES WITH PANDAS

```
# read data
data = pd.read_csv('../input/pokemon.csv')
data= data.set_index("#")
data.head()
```

Out[81]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
#											
1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
#											
4	Mega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
5	Charmander	Fire	NaN	39	52	43	60	50	65	1	False

In [82]:

```
# indexing using square brackets
data["HP"][1]
```

Out[82]:

45

In [83]:

```
# using column attribute and row label
data.HP[1]
```

Out[83]:

45

In [84]:

```
# using loc accessor
data.loc[1,["HP"]]
```

Out[84]:

```
HP    45
Name: 1, dtype: object
```

In [85]:

```
# Selecting only some columns
data[["HP","Attack"]]
```

Out[85]:

	HP	Attack
#		

	HP	Attack
#		
1	45	49
2	60	62
3	80	82
4	80	100
5	39	52
6	58	64
7	78	84
8	78	130
9	78	104
10	44	48

	HP	Attack
#		
11	59	63
12	79	83
13	79	103
14	45	30
15	50	20
16	60	45
17	40	35
18	45	25
19	65	90
20	65	150

	HP	Attack
#		
21	40	45
22	63	60
23	83	80
24	83	80
25	30	56
26	55	81
27	40	60
28	65	90
29	35	60
30	60	85

	HP	Attack
#		
...	...	...
771	95	65
772	78	92
773	67	58
774	50	50
775	45	50
776	68	75
777	90	100
778	57	80
779	43	70

	HP	Attack
#		
780	85	110
781	49	66
782	44	66
783	54	66
784	59	66
785	65	90
786	55	85
787	75	95
788	85	100
789	55	69

	HP	Attack
#		
790	95	117
791	40	30
792	85	70
793	126	131
794	126	131
795	108	100
796	50	100
797	50	160
798	80	110
799	80	160

	HP	Attack
#		
800	80	110

800 rows × 2 columns

SLICING DATA FRAME

- Difference between selecting columns
    - Series and data frames
- Slicing and indexing series
- Reverse slicing
- From something to end

In [86]:

```
# Difference between selecting columns: series and dataframes
print(type(data["HP"]))     # series
print(type(data[["HP"]]))   # data frames
<class 'pandas.core.series.Series'>
<class 'pandas.core.frame.DataFrame'>
```

In [87]:

```
# Slicing and indexing series
data.loc[1:10,"HP":"Defense"]   # 10 and "Defense" are inclusive
```

Out[87]:

	HP	Attack	Defense
#			
1	45	49	49

	HP	Attack	Defense
#			
2	60	62	63
3	80	82	83
4	80	100	123
5	39	52	43
6	58	64	58
7	78	84	78
8	78	130	111
9	78	104	78
10	44	48	65

In [88]:

```
# Reverse slicing
data.loc[10:1:-1,"HP":"Defense"]
```

Out[88]:

	HP	Attack	Defense
#			
10	44	48	65
9	78	104	78
8	78	130	111
7	78	84	78
6	58	64	58
5	39	52	43
4	80	100	123
3	80	82	83
2	60	62	63
1	45	49	49

In [89]:

```
# From something to end
data.loc[1:10,"Speed":]
```

Out[89]:

	Speed	Generation	Legendary
#			
1	45	1	False
2	60	1	False
3	80	1	False
4	80	1	False
5	65	1	False
6	80	1	False
7	100	1	False
8	100	1	False
9	100	1	False
10	43	1	False

FILTERING DATA FRAMES

Creating boolean series Combining filters Filtering column based others

In [90]:

```
# Creating boolean series
boolean = data.HP > 200
data[boolean]
```

Out[90]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
#											
122	Chansey	Normal	NaN	250	5	5	35	105	50	1	False
262	Blissey	Normal	NaN	255	10	10	75	135	55	2	False

In [91]:

```
# Combining filters
first_filter = data.HP > 150
second_filter = data.Speed > 35
data[first_filter & second_filter]
```

Out[91]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
#											
122	Chansey	Normal	NaN	250	5	5	35	105	50	1	False

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
#											
262	Blissey	Normal	NaN	255	10	10	75	135	55	2	False
352	Wailord	Water	NaN	170	90	45	90	45	60	3	False
656	Alomomola	Water	NaN	165	75	80	40	45	65	5	False

In [92]:

```
# Filtering column based others
data.HP[data.Speed<15]
```

Out[92]:

```
#
231     20
360     45
487     50
496    135
659     44
Name: HP, dtype: int64
```

TRANSFORMING DATA

- Plain python functions
- Lambda function: to apply arbitrary python function to every element
- Defining column using other columns

In [93]:

```
# Plain python functions
def div(n):
    return n/2
```

```
data.HP.apply(div)
```

Out[93]:

```
#
1      22.5
2      30.0
3      40.0
4      40.0
5      19.5
6      29.0
7      39.0
8      39.0
9      39.0
10     22.0
11     29.5
12     39.5
13     39.5
14     22.5
15     25.0
16     30.0
17     20.0
18     22.5
19     32.5
20     32.5
21     20.0
22     31.5
23     41.5
24     41.5
25     15.0
26     27.5
27     20.0
28     32.5
29     17.5
30     30.0
       ...
771    47.5
772    39.0
773    33.5
774    25.0
775    22.5
776    34.0
777    45.0
778    28.5
779    21.5
780    42.5
781    24.5
782    22.0
783    27.0
```

```
784    29.5
785    32.5
786    27.5
787    37.5
788    42.5
789    27.5
790    47.5
791    20.0
792    42.5
793    63.0
794    63.0
795    54.0
796    25.0
797    25.0
798    40.0
799    40.0
800    40.0
Name: HP, Length: 800, dtype: float64
```

In [94]:

```python
# Or we can use lambda function
data.HP.apply(lambda n : n/2)
```

Out[94]:

```
#
1      22.5
2      30.0
3      40.0
4      40.0
5      19.5
6      29.0
7      39.0
8      39.0
9      39.0
10     22.0
11     29.5
12     39.5
13     39.5
14     22.5
15     25.0
16     30.0
17     20.0
18     22.5
19     32.5
20     32.5
21     20.0
22     31.5
23     41.5
24     41.5
```

```
25     15.0
26     27.5
27     20.0
28     32.5
29     17.5
30     30.0
       ...
771    47.5
772    39.0
773    33.5
774    25.0
775    22.5
776    34.0
777    45.0
778    28.5
779    21.5
780    42.5
781    24.5
782    22.0
783    27.0
784    29.5
785    32.5
786    27.5
787    37.5
788    42.5
789    27.5
790    47.5
791    20.0
792    42.5
793    63.0
794    63.0
795    54.0
796    25.0
797    25.0
798    40.0
799    40.0
800    40.0
Name: HP, Length: 800, dtype: float64
```

In [95]:

```
# Defining column using other columns
data["total_power"] = data.Attack + data.Defense
data.head()
```

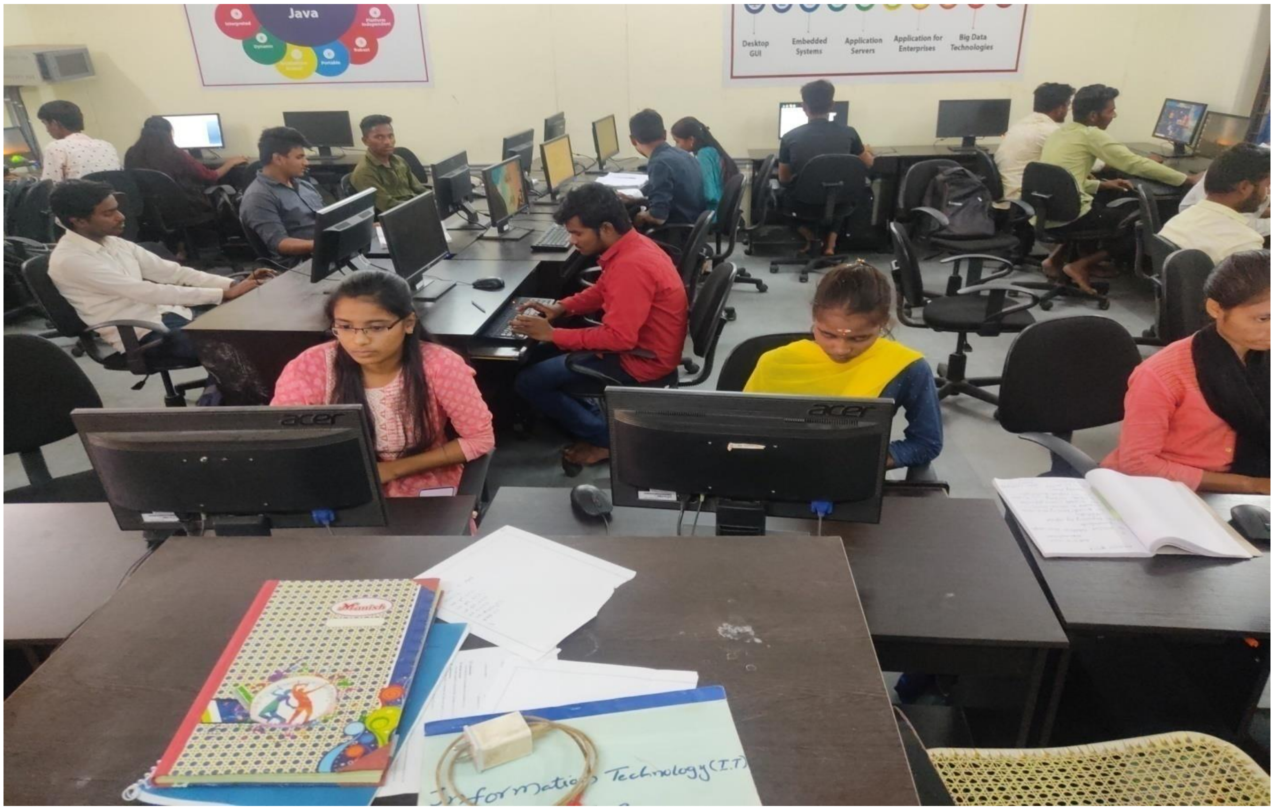
Out[95]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary	total_power
#												
1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False	98
2	Ivysaur	Grass	Poison	60	62	63	80	80				

PHOTOS


LIST OF COMPUTER PROGRAMMING LANGUAGES


James Gosling
SURYA


Java
Desktop GUI
Embedded Systems
Application Servers
Application for Enterprises
Big Data Technologies
acer
acer
Information Technology (I.T)

Attendance

Class : BCom-CA - III - C 2022-23
I sem

KAKATIYA GOVERNMENT COLLEGE
Hands-on Workshop on Python Libraries for Machine Learning

KAKATIYA GOVT. COLLEGE
College Code : 006
Hanamkonda, Warangal

SNo	HTNo	Name	08/09/22	12/09/22	13/09/22	14/09/22	15/09/22	16/09/22	17/09/22	19/09/22	20/09/22	21/09/22
1	006212180	KOMMULA NITHIN	K. Nithin	K. Nithin	K. Nithin	K. Nithin	K. Nithin	K. Nithin	K. Nithin	K. Nithin	K. Nithin	K. Nithin
2	006212181	KOMMULA NITHIN	K. Nithin	K. Nithin	Nithin	Nithin	Nithin	Nithin	Nithin	Nithin	Nithin	Nithin
3	006212182	KONDAPALLY VENKATESH	[illegible]	[illegible]	K. Venkatesh	Venkatesh	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	006212183	KONGARI PRUDHVI RAJ	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj	K. Prudhvi Raj
5	006212184	KOPPULA HARI KRISHNA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	006212185	KOPPULA NAGARAJU	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj
7	006212186	KORUKOPPULA THARUN	Tharun	Tharun	Tharun	Tharun	Tharun	Tharun	Tharun	Tharun	Tharun	Tharun
8	006212187	KOTHAPALLI SUNNY	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny
9	006212188	KOUDAGANI ANIL	Anil	Anil	Anil	Anil	Anil	Anil	Anil	Anil	Anil	Anil
10	006212189	KOULAGONI VINAY	Vinay	Vinay	Vinay	Vinay	Vinay	Vinay	Vinay	Vinay	Vinay	Vinay
11	006212190	KOWDAGANI TEJA	K. Teja	K. Teja	K. Teja	K. Teja	K. Teja	K. Teja	K. Teja	K. Teja	K. Teja	K. Teja
12	006212191	KOYYADA VAMSHI	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi
13	006212192	KUMMARI MAHENDAR	K. Mahendar	K. Mahendar	K. Mahendar	K. Mahendar	K. Mahendar	K. Mahendar	K. Mahendar	[illegible]	[illegible]	[illegible]
14	006212193	KUNUSOTH NAVEEN	K. Naveen	K. Naveen	K. Naveen	K. Naveen	K. Naveen	K. Naveen	K. Naveen	K. Naveen	K. Naveen	K. Naveen
15	006212194	KURRI MOUNIKA	K. Mounika	K. Mounika	K. Mounika	K. Mounika	K. Mounika	K. Mounika	K. Mounika	K. Mounika	K. Mounika	K. Mounika
16	006212195	KURSAM YASHODA										
17	006212196	LAKKARSU SAI	Sai	Sai	Sai	Sai	Sai	Sai	Sai	Sai		
18	006212197	LAVUDYA NAVYA	Navya	Navya	Navya	Navya	Navya	Navya	Navya	Navya	Navya	Navya
19	006212198	LAVUDYA THIRUPATHI	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
20	006212199	LINGALA SPANDANA	Spandana	Spandana	Spandana	Spandana	Spandana	Spandana	Spandana	Spandana	Spandana	Spandana
21	006212200	LODE SHRAVAN KUMAR	L. Sravan	L. Sravan	L. Sravan	L. Sravan	L. Sravan	L. Sravan	L. Sravan	L. Sravan	L. Sravan	L. Sravan
22	006212201	LOTTI POOJITHA	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha
23	006212202	LYADALLA KEERTHI	L. Keerthi	L. Keerthi	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
24	006212203	MADDEBOINA PRASHANTH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Prash	Prash	[illegible]	[illegible]	[illegible]
25	006212204	MADI SHASHI VARDHAN	M. Shashivardhan	M. Shashivardhan	M. Shashivardhan	Shashi	Shashi	Shashi	M. S	M. S	M. S	M. S
26	006212205	MAHANKALI NAGARAJU	M. Nagaraj	M. Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj	Nagaraj
27	006212206	MALLABOINA RAKESH	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh	M. Rakesh
28	006212207	MALOTH THIRUPATHI	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
29	006212208	MAMIDI SRAVAN KUMAR	Sravan	Sravan	Sravan	Sravan	Sravan	Sravan	Sravan	Sravan	Sravan	Sravan
30	006212209	MANCHALA RAJU	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
31	006212210	MANDA CHANDU	Chandu	Chandu	Chandu	Chandu	Chandu	Chandu	Chandu	Chandu	Chandu	Chandu
32	006212211	MANDHA VAMSHI	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi
33	006212212	MANKURTHI SURESH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
34	006212213	MANTHURTHI RAKESH	Rakesh	Rakesh	Rakesh	Rakesh	Rakesh	Rakesh	Rakesh	Rakesh	Rakesh	Rakesh
35	006212214	MANUPATI RAJASHEKER	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
36	006212215	MARAPALLI ASHISH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]