TRF Robospark
Final Task - Report

1.Project Name -: Predict the grades

2.Group Information -: Group 1

Member 1-:

- Name - Rutuja Khaire
- Email id -: rutuja.khaire19@vit.edu
- Gr No. -: 11910769
- Year. - Third year

Member 2-:

- Name -: Varad Ingale
- Email id -: varad.ingale20@vit.edu
- Pr No -: 12010217
- Year -: Second year

Member 3-:

- Name -: Harshit Mundhra
- Email id -: harshit.mundhra20@vit.edu
- Pr no. -: 12010752
- Year -: Second year

3. Project GitHub Link -:
https://github.com/rohitsingh0210vit/Robospark-2021-FT-predict-the-grades

4. Project Algorithm/Workflow -:

Phase 1 -:

- Importing Dataset
- Data Preprocessing
- Data Visualization
- Encoding -:

    - Label Encoding
    - One hot encoding
- Training the Dataset
- Three algorithmic comparative Analysis -:
    - a).Linear Regression -:
    - b).Decision Tree
    - c).Random Forest
- Score Prediction

Phase 2 -:

- Hyper parameter Tuning for Random Forest Algorithm -:
    - Randomized Search CV
    - Grid Search CV

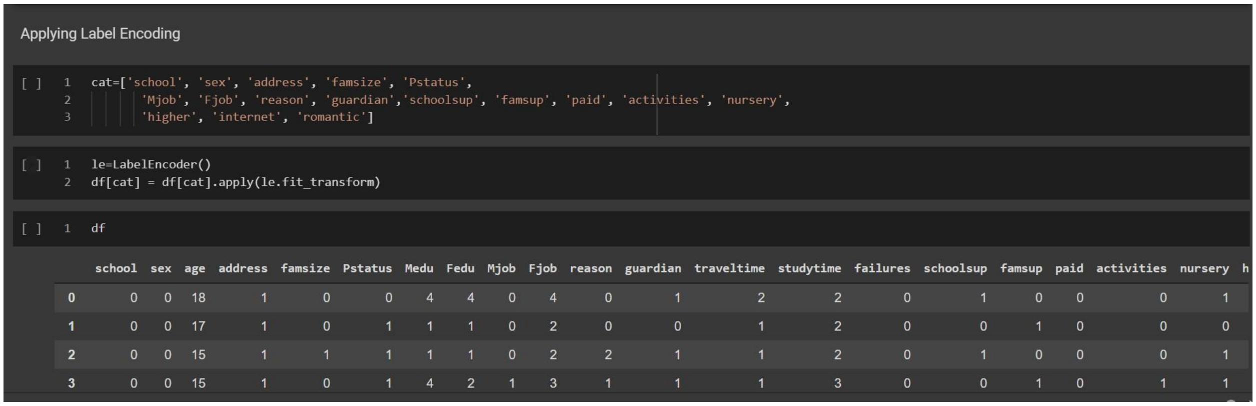
5. Problems Faced -:

- Confusions in the different techniques of encoding.
- After trying Classification, Score came up to be less.
- Not sure about Parameters for tuning due to fluctuations in Score.

6. Alternative Solutions found for problems mentioned above -:

- One Hot encoding and Label Encoding were used to ensure best Accuracy and Scores.
- Instead of classification, regression technique is used for better accuracy.
- Trial and Error method was used for changing hyper parameters of Random Forest Algorithm.

7. Code snippet and Output ss -:



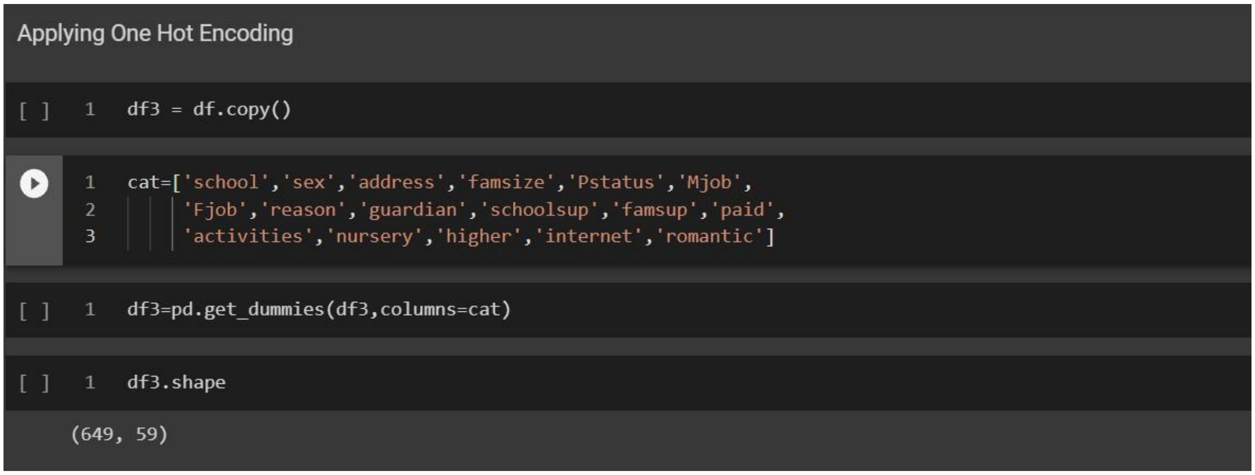
Applying Label Encoding

```
cat=['school', 'sex', 'address', 'famsize', 'Pstatus',
     'Mjob', 'Fjob', 'reason', 'guardian','schoolsup', 'famsup', 'paid', 'activities', 'nursery',
     'higher', 'internet', 'romantic']
```

```
le=LabelEncoder()
df[cat] = df[cat].apply(le.fit_transform)
```

```
df
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	traveltime	studytime	failures	schoolsup	famsup	paid	activities	nursery	h
0	0	0	18	1	0	0	4	4	0	4	0	1	2	2	0	1	0	0	0	1	
1	0	0	17	1	0	1	1	1	0	2	0	0	1	2	0	0	1	0	0	0	
2	0	0	15	1	1	1	1	1	0	2	2	1	1	2	0	1	0	0	0	1	
3	0	0	15	1	0	1	4	2	1	3	1	1	1	3	0	0	1	0	1	1	



Applying One Hot Encoding

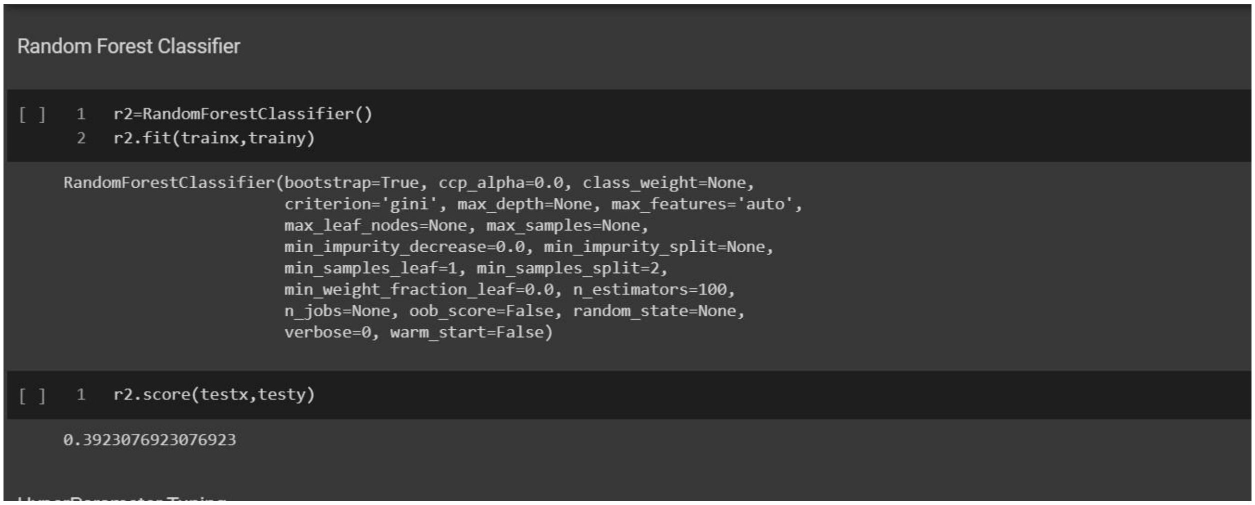
```
df3 = df.copy()
```

```
cat=['school','sex','address','famsize','Pstatus','Mjob',
     'Fjob','reason','guardian','schoolsup','famsup','paid',
     'activities','nursery','higher','internet','romantic']
```

```
df3=pd.get_dummies(df3,columns=cat)
```

```
df3.shape
```

```
(649, 59)
```



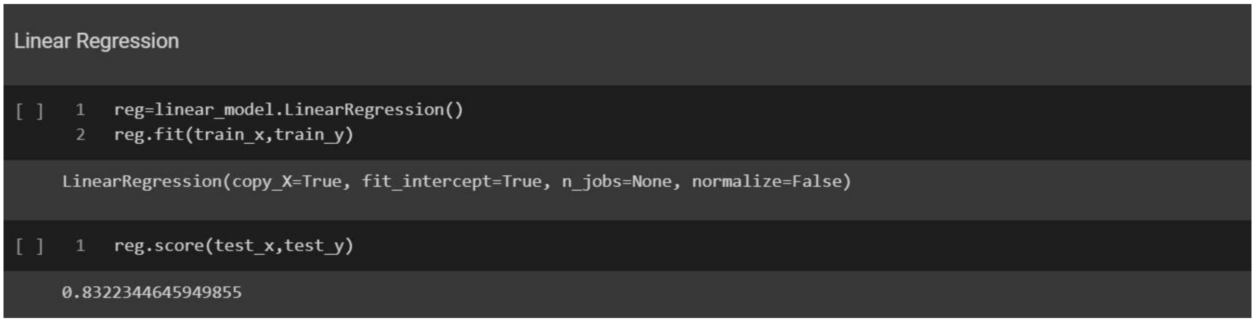
Random Forest Classifier

```
r2=RandomForestClassifier()
r2.fit(trainx,trainy)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
r2.score(testx,testy)
```

```
0.3923076923076923
```



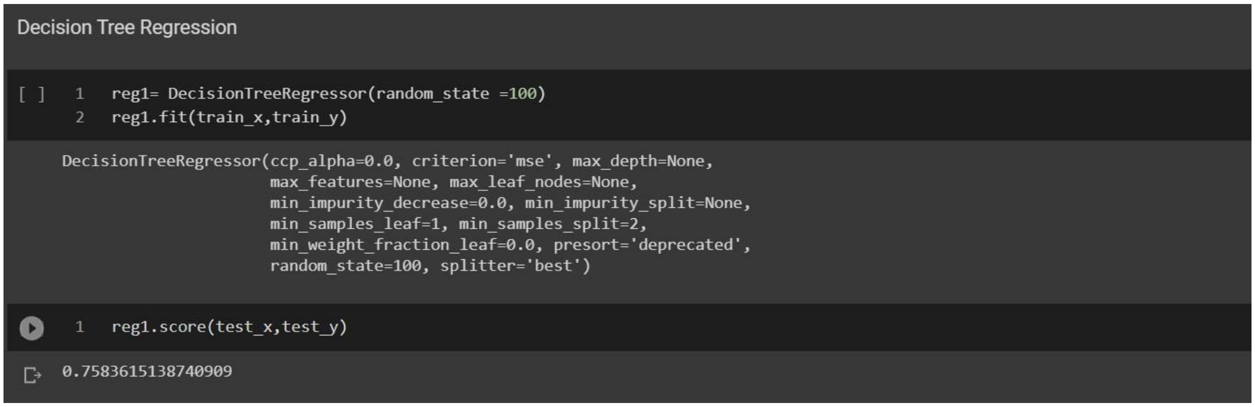
Linear Regression

```
reg=linear_model.LinearRegression()
reg.fit(train_x,train_y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
reg.score(test_x,test_y)
```

```
0.8322344645949855
```



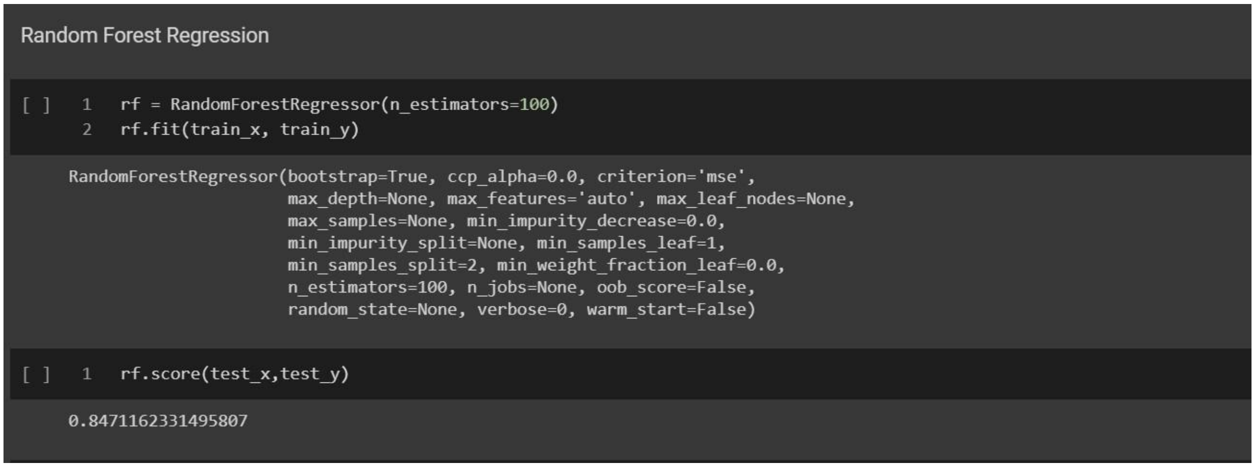
Decision Tree Regression

```
reg1= DecisionTreeRegressor(random_state =100)
reg1.fit(train_x,train_y)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', 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='deprecated',
                      random_state=100, splitter='best')
```

```
reg1.score(test_x,test_y)
```

```
0.7583615138740909
```



Random Forest Regression

```
rf = RandomForestRegressor(n_estimators=100)
rf.fit(train_x, train_y)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=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=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
rf.score(test_x,test_y)
```

```
0.8471162331495807
```