Delivery of Sprint – 1

Team Id: PNT2022TMID21248

In this Sprint, we have built a ML model for project using Random Forest.

Source Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

dataset=pd.read_csv("loan1.csv")

dataset.head()

dataset.isnull()

d1=dataset.fillna(method='bfill')

print(d1['Credit_History'])

x=d1[['Gender','Married','Dependents','Education','Self_Employed','ApplicantIncome','Coapplicant_Income','LoanAmount','Loan_Amount_Term','Credit_History','Emi']]

y=d1['Alloted_amount']

print(x)

df_dummies = pd.get_dummies(x, prefix='', prefix_sep='', columns=['Gender', 'Married','Education','Self_Employed'])
```

```
x=x.drop(['Dependents', 'ApplicantIncome', 'Coapplicant_Income',
'LoanAmount', 'Loan_Amount_Term', 'Credit_History',
'Emi','Gender','Married','Education','Self_Employed'],axis=1)

x=pd.concat([x,df_dummies],axis=1)

print(list(x.columns))

fea_list=list(x.columns)

print(fea_list)


from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=50
)

#print(x_test)

from sklearn.linear_model import LinearRegression

reg=LinearRegression()

reg.fit(x_train,y_train)

y_pred=reg.predict(x_test)

print("Predicted Values:")

print(y_pred)

#p=reg.predict([0,24870,0,86700,18,1,966.777,200000,18,1,1000])

#print(p)

print("Regression Coefficient:",reg.coef_)

print("Regression Intercept:",reg.intercept_)

print("Regression Score:",reg.score(x_train,y_train))


from sklearn.ensemble import RandomForestRegressor

# Instantiate model with 1000 decision trees

rf = RandomForestRegressor(n_estimators = 1000, random_state=50)

# Train the model on training data
```

```
rf.fit(x_train,y_train);

predictions=rf.predict(x_test)
# Calculate the absolute errors
errors = abs(predictions -y_test)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / y_test)
# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

importances = list(rf.feature_importances_)

# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 5)) for feature, importance
in zip(fea_list, importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1],
reverse = True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
feature_importances];
from sklearn.metrics import mean_squared_error
# Set the style
plt.style.use('fivethirtyeight')
```

```
# list of x locations for plotting
x_values = list(range(len(importances)))
# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical')
# Tick labels for x axis
plt.xticks(x_values, fea_list, rotation='vertical')
# Axis labels and title
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');

#prediction using Random Forest
print(rf.predict([[0,50000,42000,200000,60,0,3000,0,1,0,1,0,1,1,0]]))
print("root_mean_sqrd_error is=",np.sqrt(mean_squared_error(y_test,predictions)))

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
# predicting the accuracy score
score=r2_score(y_test,y_pred)
print("r2 score is ",score)
print("mean_sqrd_error is=",mean_squared_error(y_test,y_pred))
print("root_mean_squared error of is=",np.sqrt(mean_squared_error(y_test,y_pred)))

plt.plot(x_train,y_train,color="r",marker="*",markersize=15)
plt.plot(x_test,y_test,color="b",marker="*",markersize=15)
plt.show()
```

#['Dependents', 'ApplicantIncome', 'Coapplicant_Income', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Emi', 'Female', 'Male', 'No', 'Yes', 'Graduate', 'Not Graduate', 'No', 'Yes']

print(reg.predict([[0,50000,42000,200000,60,0,3000,0,1,0,1,0,1,1,0]]))

Result Screenshots:

Dataset Details:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicant_Income	LoanAmount	Loan	Loan_Amount_Term	Cred
0	LP001002	Male	No	0.0	Graduate	No	52690	0.0	100000	100000.0	12	
1	LP001003	Male	Yes	1.0	Graduate	No	45830	15080.0	18000	18000.0	18	
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	85000	85000.0	18	
3	LP001006	Male	Yes	0.0	Not Graduate	No	25830	23580.0	100000	100000.0	18	
4	LP001008	Male	No	0.0	Graduate	No	60000	0.0	150000	150000.0	18	

Regression info:

```
Regression Coefficient: [-1.39805695e+01  1.49961042e+00  1.49969645e+00 -7.96100193e-06
 -5.22027611e-01  1.26749599e+01 -1.74532309e-03 -7.52720260e+00
  7.52720260e+00 -1.36994649e+01  1.36994649e+01  1.36252738e+01
 -1.36252738e+01 -6.32886295e+01  6.32886295e+01]
Regression Intercept: 84.07628588609805
Regression Score: 0.999988213585471
```

Validation:

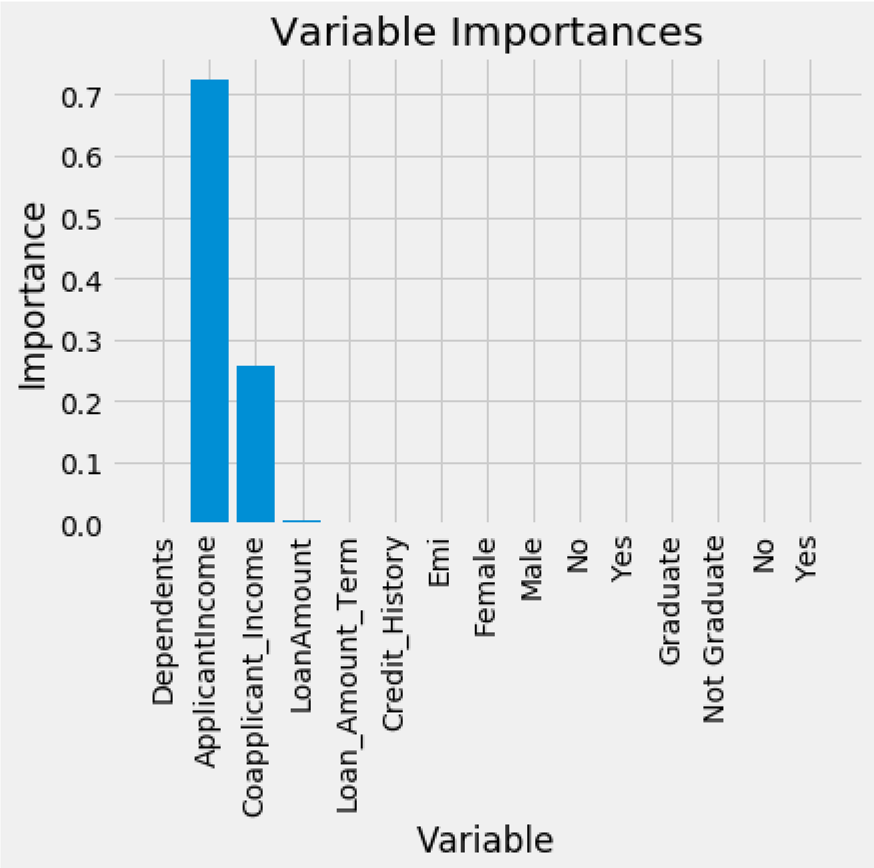
```
Mean Absolute Error: 6840.34 degrees.
Accuracy: 95.39 %.
```

Importance of independent variables:

```
Variable: ApplicantIncome        Importance: 0.72561
Variable: Coapplicant_Income     Importance: 0.25648
Variable: LoanAmount             Importance: 0.00396
Variable: Emi                    Importance: 0.00352
Variable: Loan_Amount_Term       Importance: 0.00251
Variable: Dependents             Importance: 0.00246
Variable: Yes                    Importance: 0.00116
Variable: Female                 Importance: 0.00107
Variable: Male                   Importance: 0.00088
Variable: No                     Importance: 0.00077
Variable: Yes                    Importance: 0.0007
Variable: No                     Importance: 0.00065
Variable: Credit_History         Importance: 0.00017
Variable: Graduate               Importance: 3e-05
Variable: Not Graduate           Importance: 3e-05
```

Linear regression predicted values:

```
[136142.28]
root_mean_sqrd_error is= 42032.81981282777
```



Random forest regressor validation:

```
r2 score is  0.9999996538205425
mean_sqrd_error is= 3399.8105132165288
root_mean_squared error of is= 58.307894090050354
```

Random Forest Output:

```
#['Dependents', 'ApplicantIncome', 'Coapplicant_Income', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Emi', 'Female', 'Mc
print(reg.predict([[0,50000,42000,200000,60,0,3000,0,1,0,1,0,1,1,0]]))
```

```
[137958.01084732]
```