PREDECTING PERSONAL LOAN APPROVAL

USING

MACHINE LEARNING

TEAM SIZE:4

TEAM LEADER: **1.SRIDHAR.P(20UCS2515)**

TEAM MEMBERS: **1.SURENDHAR.R(20UCS2516)**
2.SEENIVASAN.R(20UCS2513)
3.SILAMBARASAN.V(20UCS2514)

INTRODUCTION

OVERVIEW

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages,auto loans, student loans, business loans and many more.They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees,vary depending on the lender and the type of loan.A personal loan is a type of unsecured loan that can be used for a variety of expenses such as homerepairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's creditworthiness.To qualify for a personal loan,borrowers typically need to provide proof of income and have a good credit score. redicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

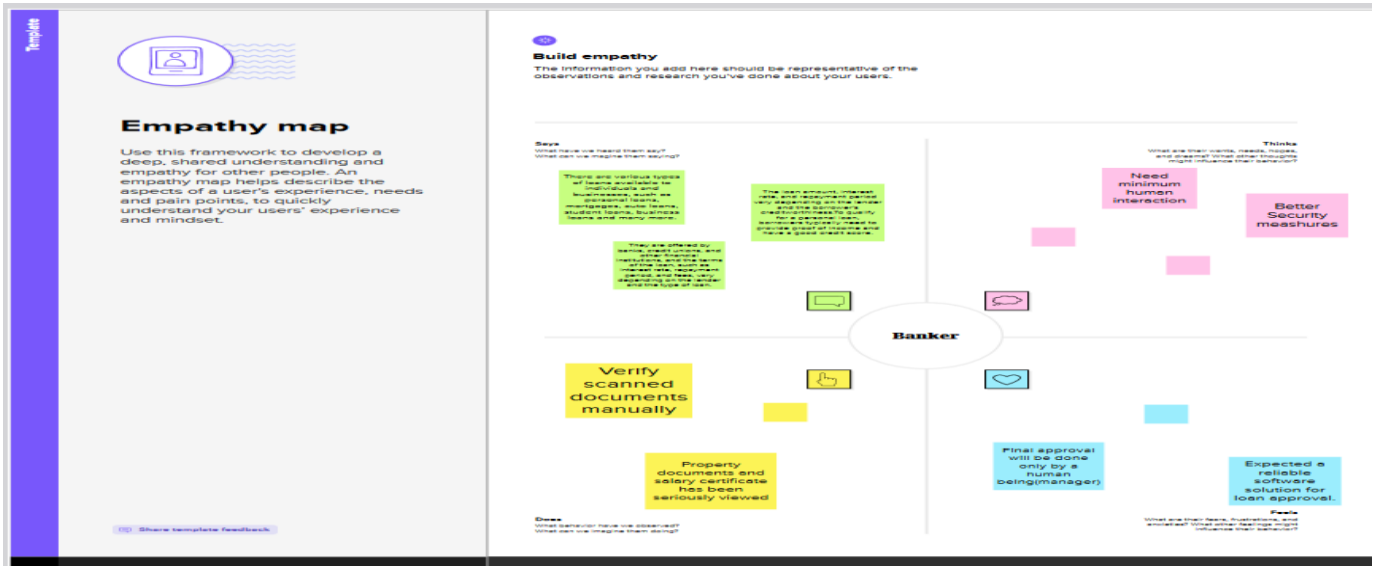
PURPOSE

Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim of this

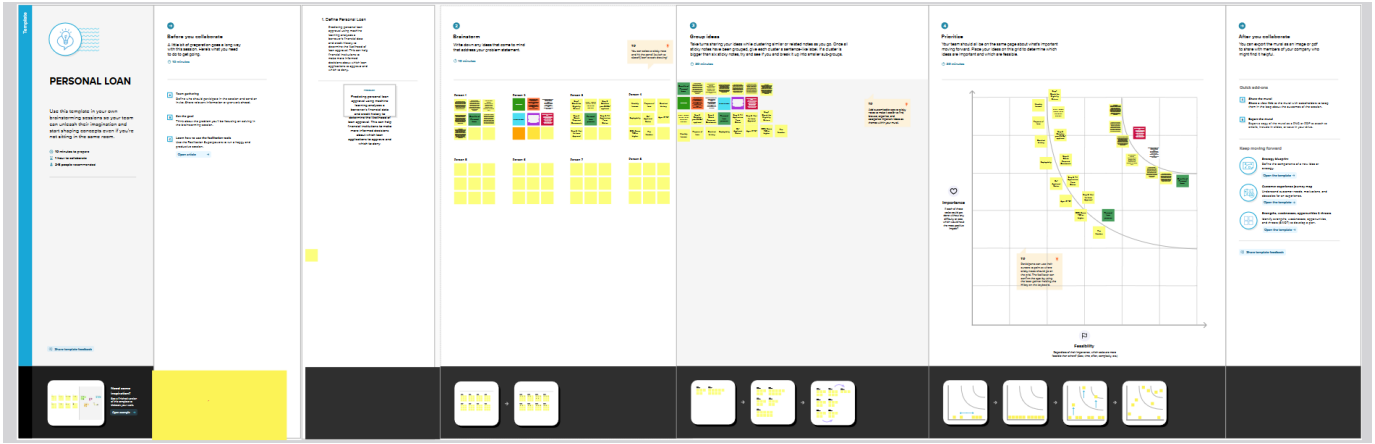
Paper is to provide quick, immediate and easy way to choose the deserving applicants**.** Dream housing Finance Company deals in all loans. They have presence across all urban, semi urban and rural areas. Customer first apply for loan after that company or bank validates the customer eligibility for loan. Company or bank wants to automate the loan eligibility process (real time) based on customer details provided while filling application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and other. This project has taken the data of previous customers of various banks to whom on a set of parameters loan were approved. So the machine learning model is trained on that record to get accurate results. Our main objective of this project is to predict the safety of loan. To predict loan safety, the SVM and Naïve bayes algorithm are used. First the data is cleaned so as to avoid the missing values in the data set.

PROBLEM DEFINING AND DESIGN THINKING

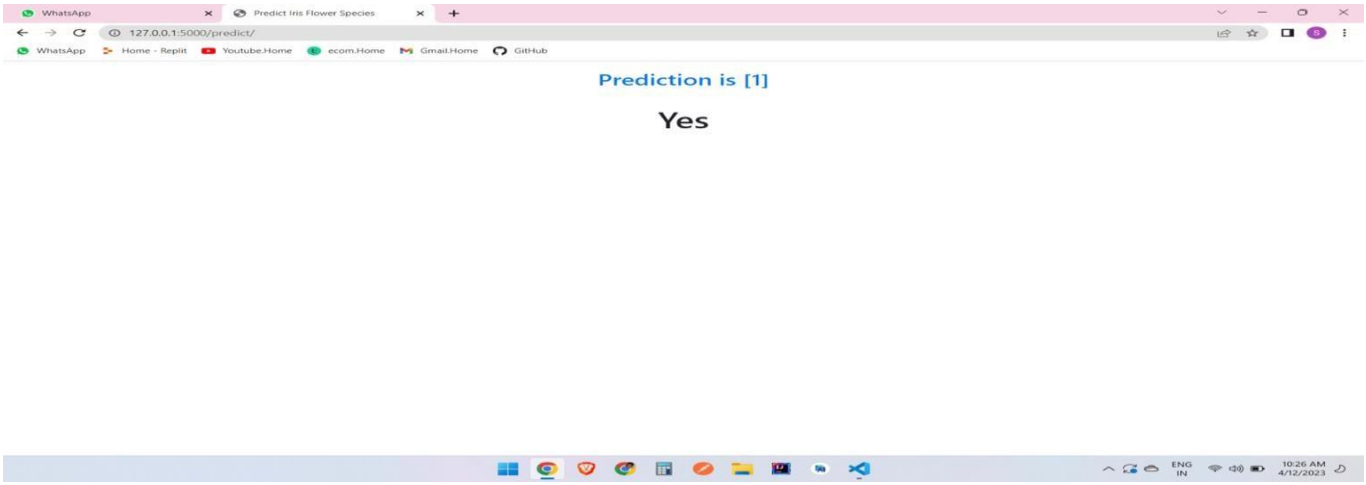
EMPATHY MAP

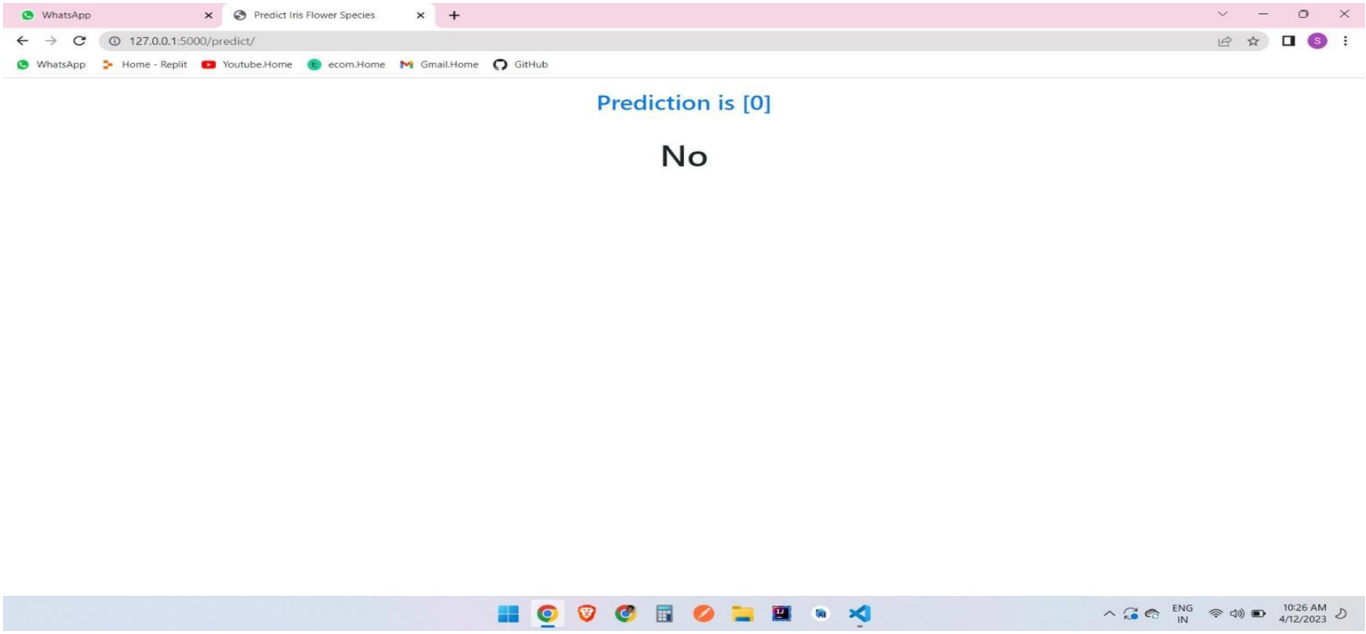


IDEATION &BRAINSTROMING MAP



RESULT


WhatsApp
Predict Iris Flower Species
127.0.0.1:5000/predict/
WhatsApp
Home - Replit
Youtube.Home
ecom.Home
Gmail.Home
GitHub
Prediction is [1]
Yes
ENG
IN
10:26 AM
4/12/2023


WhatsApp
Predict Iris Flower Species
127.0.0.1:5000/predict/
WhatsApp
Home - Replit
Youtube.Home
ecom.Home
Gmail.Home
GitHub
Prediction is [0]
No
ENG
IN
10:26 AM
4/12/2023

ADVANTAGES

Accuracy—one of the primary benefits of using machine learning for credit scoring is its accuracy. Unlike human manual processing, ML-based models are automated and less likely to make mistakes. This means that loan processing becomes not only faster but more accurate, too, cutting costs on the whole.

It is done by predicting if the loan can be given to that person on the basis of various parameters like credit score, income, age, marital status, gender, etc. The prediction model not only helps the applicant but also helps the bank by minimizing the risk and reducing the number of defaulters.

Using historical data, machine learning algorithms can learn to identify trends and patterns to evaluate possible outcomes. Using this as its reference framework, the algorithm can repeat the process with current data to make predictions about the future

Machine learning allows banks to proactively monitor customer behavior, identify anomalies in real time, reduce the probability of false positives, and prevent fraud. Machine learning's ability to model how a bank will react to certain economic conditions allows decisionmakers to create more informed strategies.

DISADVANTAGES

The disadvantage of this model is that it emphasize different weights to each factor but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system.

Loans are not very flexible - you could be paying interest on funds you're not using. You could have trouble making monthly repayments if your customers don't pay you promptly, causing cashflow problems

Strict eligibility criteria. One of the major disadvantages of a bank loan is that banks can be cautious about lending to small businesses.

Inaccuracies are a common occurrence since algorithms and can sometimes be underdeveloped. Indeed human is to error, and since a human makes these algorithms, mistakes in the coding may be overlooked

Risk of hacking and scams- Be careful about the data you share and the online loan apps you use, as these can be prone to data hacking since the whole loan process is done online.

APPLICATION

Loan Approval System is software used for approval of loan in banking sector. In this proposed system we have used machine learning algorithm. Machine Learning is process in which a symmetric model is build from the existing dataset; this model is applied for the testing of the new dataset.

It is done by predicting if the loan can be given to that person on the basis of various parameters like credit score, income, age, marital status, gender, etc. The prediction model not only helps the applicant but also helps the bank by minimizing the risk and reducing the number of defaulters.

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more.

CONCLUSION

From the proper view of analysis this system can be used perfect for detection of clients who are eligible for approval of loan. The software is working perfect and can be used for all banking requirements. This system can be easily uploaded in any operating system. Since the technology is moving towards online, this system has more scope for the upcoming days. This system is more secure and reliable. Since we have used Random Forest Algorithm the system returns very accurate results. There is no issue if there are many no of customers applying for loan. This system accepts data for N no. of customers. In future we can add more algorithms to this system for getting more accurate results.

FURTHER SCOPE

In this project we use the float data type for in the data set in further update we can use string data type

REFERENCE

[1] K. Hanumantha Rao., G. Srinivas., A. Damodhar., M.Vikas Krishna., Implementation of Anomaly Detection Technique Using Machine Learning Algorithms: International Journal of Computer Science and Telecommunications, Vol. 2, Issue 3, 2011.

[2] S.S. Keerthi., E.G. Gilbert., Convergence of a generalize SMO algorithm for SVM classifier design, Machine Learning, Springer, Vol. 4, Issue 1, pp. 351-360, 2002.

[3] Andy Liaw., Matthew Wiener., Classification and Regression by random Forest, Vol. 2, Issue 3, pp. 9-22,2002

[4] SOURCE CODE LINK: https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problemdataset

APPENDIX

SOURCE CODE

```
import pandas as pd

train_df = pd.read_csv(r'C:\Users\sridh\OneDrive\Desktop\new
project\dataset\train_u6lujuX_CVtuZ9i.csv')
train_df.info()


train_df = train_df.drop(columns=['Loan_ID']) ## Dropping Loan ID
categorical_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
'Property_Area','Credit_History','Loan_Amount_Term']
#categorical_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
'Property_Area','Loan_Amount_Term']

print(categorical_columns)
numerical_columns = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']
print(numerical_columns)



import seaborn as sns
import matplotlib.pyplot as plt


fig,axes = plt.subplots(4,2,figsize=(12,15))
for idx,cat_col in enumerate(categorical_columns):
    row,col = idx//2,idx%2
    sns.countplot(x=cat_col,data=train_df,hue='Loan_Status',ax=axes[row,col])


plt.subplots_adjust(hspace=1)




fig,axes = plt.subplots(1,3,figsize=(17,5))
for idx,cat_col in enumerate(numerical_columns):
    sns.boxplot(y=cat_col,data=train_df,x='Loan_Status',ax=axes[idx])

print(train_df[numerical_columns].describe())
plt.subplots_adjust(hspace=1)


train_df_encoded = pd.get_dummies(train_df,drop_first=True)
train_df_encoded.head()
```

```
train_df_encoded = pd.get_dummies(train_df,drop_first=True)
train_df_encoded.head()


########## Split Features and Target Varible ############
X = train_df_encoded.drop(columns='Loan_Status_Y')
y = train_df_encoded['Loan_Status_Y']

################# Splitting into Train -Test Data #######
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,stratify
=y,random_state =42)
############### Handling/Imputing Missing values #############
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy='mean')
imp_train = imp.fit(X_train)
X_train = imp_train.transform(X_train)
X_test_imp = imp_train.transform(X_test)



from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score,f1_score


tree_clf = DecisionTreeClassifier()
tree_clf.fit(X_train,y_train)
y_pred = tree_clf.predict(X_train)
print("Training Data Set Accuracy: ", accuracy_score(y_train,y_pred))
print("Training Data F1 Score ", f1_score(y_train,y_pred))

print("Validation Mean F1 Score:
",cross_val_score(tree_clf,X_train,y_train,cv=5,scoring='f1_macro').mean())
print("Validation Mean Accuracy:
",cross_val_score(tree_clf,X_train,y_train,cv=5,scoring='accuracy').mean())




training_accuracy = []
val_accuracy = []
training_f1 = []
val_f1 = []
tree_depths = []
```

```
for depth in range(1,20):
    tree_clf = DecisionTreeClassifier(max_depth=depth)
    tree_clf.fit(X_train,y_train)
    y_training_pred = tree_clf.predict(X_train)

    training_acc = accuracy_score(y_train,y_training_pred)
    train_f1 = f1_score(y_train,y_training_pred)
    val_mean_f1 = cross_val_score(tree_clf,X_train,y_train,cv=5,scoring='f1_macro').mean()
    val_mean_accuracy =
cross_val_score(tree_clf,X_train,y_train,cv=5,scoring='accuracy').mean()

    training_accuracy.append(training_acc)
    val_accuracy.append(val_mean_accuracy)
    training_f1.append(train_f1)
    val_f1.append(val_mean_f1)
    tree_depths.append(depth)


Tuning_Max_depth = {"Training Accuracy": training_accuracy, "Validation Accuracy":
val_accuracy, "Training F1": training_f1, "Validation F1":val_f1, "Max_Depth": tree_depths
}
Tuning_Max_depth_df = pd.DataFrame.from_dict(Tuning_Max_depth)

plot_df = Tuning_Max_depth_df.melt('Max_Depth',var_name='Metrics',value_name="Values")
fig,ax = plt.subplots(figsize=(15,5))
sns.pointplot(x="Max_Depth", y="Values",hue="Metrics", data=plot_df,ax=ax)



import graphviz
from sklearn import tree

tree_clf = tree.DecisionTreeClassifier(max_depth = 3)
tree_clf.fit(X_train,y_train)
dot_data = tree.export_graphviz(tree_clf,feature_names = X.columns.tolist())
graph = graphviz.Source('dot_data')
graph



training_accuracy = []
val_accuracy = []
training_f1 = []
val_f1 = []
min_samples_leaf = []
import numpy as np
for samples_leaf in range(1,80,3): ### Sweeping from 1% samples to 10% samples per leaf
    tree_clf = DecisionTreeClassifier(max_depth=3,min_samples_leaf = samples_leaf)
    tree_clf.fit(X_train,y_train)
```

```
    y_training_pred = tree_clf.predict(X_train)

    training_acc = accuracy_score(y_train,y_training_pred)
    train_f1 = f1_score(y_train,y_training_pred)
    val_mean_f1 = cross_val_score(tree_clf,X_train,y_train,cv=5,scoring='f1_macro').mean()
    val_mean_accuracy =
cross_val_score(tree_clf,X_train,y_train,cv=5,scoring='accuracy').mean()

    training_accuracy.append(training_acc)
    val_accuracy.append(val_mean_accuracy)
    training_f1.append(train_f1)
    val_f1.append(val_mean_f1)
    min_samples_leaf.append(samples_leaf)


Tuning_min_samples_leaf = {"Training Accuracy": training_accuracy, "Validation Accuracy":
val_accuracy, "Training F1": training_f1, "Validation F1":val_f1, "Min_Samples_leaf":
min_samples_leaf }
Tuning_min_samples_leaf_df = pd.DataFrame.from_dict(Tuning_min_samples_leaf)

plot_df =
Tuning_min_samples_leaf_df.melt('Min_Samples_leaf',var_name='Metrics',value_name="Values")
fig,ax = plt.subplots(figsize=(15,5))
sns.pointplot(x="Min_Samples_leaf", y="Values",hue="Metrics", data=plot_df,ax=ax)




from sklearn.metrics import confusion_matrix
tree_clf = DecisionTreeClassifier(max_depth=3,min_samples_leaf = 35)
tree_clf.fit(X_train,y_train)
y_pred = tree_clf.predict(X_test_imp)
print("Test Accuracy: ",accuracy_score(y_test,y_pred))
print("Test F1 Score: ",f1_score(y_test,y_pred))
print("Confusion Matrix on Test Data")
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)


from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(n_estimators=100,max_depth=3,min_samples_leaf = 10)
rf_clf.fit(X_train,y_train)
y_pred = rf_clf.predict(X_train)
print("Train F1 Score ", f1_score(y_train,y_pred))
print("Train Accuracy ", accuracy_score(y_train,y_pred))

print("Validation Mean F1 Score:
",cross_val_score(rf_clf,X_train,y_train,cv=5,scoring='f1_macro').mean())
```

```
print("Validation Mean Accuracy:
",cross_val_score(rf_clf,X_train,y_train,cv=5,scoring='accuracy').mean())

y_pred = rf_clf.predict(X_test_imp)
print("Test Accuracy: ",accuracy_score(y_test,y_pred))
print("Test F1 Score: ",f1_score(y_test,y_pred))
print("Confusion Matrix on Test Data")
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)


from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_predict

train_accuracies = []
train_f1_scores = []
test_accuracies = []
test_f1_scores = []
thresholds = []

#for thresh in np.linspace(0.1,0.9,8): ## Sweeping from threshold of 0.1 to 0.9
for thresh in np.arange(0.1,0.9,0.1): ## Sweeping from threshold of 0.1 to 0.9
    logreg_clf = LogisticRegression(solver='liblinear')
    logreg_clf.fit(X_train,y_train)

    y_pred_train_thresh = logreg_clf.predict_proba(X_train)[:,1]
    y_pred_train = (y_pred_train_thresh > thresh).astype(int)

    train_acc = accuracy_score(y_train,y_pred_train)
    train_f1 = f1_score(y_train,y_pred_train)

    y_pred_test_thresh = logreg_clf.predict_proba(X_test_imp)[:,1]
    y_pred_test = (y_pred_test_thresh > thresh).astype(int)

    test_acc = accuracy_score(y_test,y_pred_test)
    test_f1 = f1_score(y_test,y_pred_test)

    train_accuracies.append(train_acc)
    train_f1_scores.append(train_f1)
    test_accuracies.append(test_acc)
    test_f1_scores.append(test_f1)
    thresholds.append(thresh)


Threshold_logreg = {"Training Accuracy": train_accuracies, "Test Accuracy":
test_accuracies, "Training F1": train_f1_scores, "Test F1":test_f1_scores, "Decision
Threshold": thresholds }
Threshold_logreg_df = pd.DataFrame.from_dict(Threshold_logreg)
```

```
plot_df = Threshold_logreg_df.melt('Decision
Threshold',var_name='Metrics',value_name="Values")
fig,ax = plt.subplots(figsize=(15,5))
sns.pointplot(x="Decision Threshold", y="Values",hue="Metrics", data=plot_df,ax=ax)

thresh = 0.4 ### Threshold chosen from above Curves
y_pred_test_thresh = logreg_clf.predict_proba(X_test_imp)[:,1]
y_pred = (y_pred_test_thresh > thresh).astype(int)
print("Test Accuracy: ",accuracy_score(y_test,y_pred))
print("Test F1 Score: ",f1_score(y_test,y_pred))
print("Confusion Matrix on Test Data")
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
```