Predicting Home Loan Approval

By: Taji Abdullah, Mark Werden, Gustavo Gyotoku, Sammy Mohn, Elza Hayyat

Why is This Topic Important?

- Real estate is socioeconomically transformative
- Property ownership is the cornerstone to economic success in the U.S
- One of the best ways to build generational wealth
- Simply put can positively change you and your family's life

Traditional Home Loan Applications

- Details of the purchase contract and the property
- Personal information, including SSN
- Employment History
- Verification of Employment (VOE)
- Information on other income
- Copies of documents showing money owed to the borrower(s)
- Information on the ethnicity, race, and gender of the applicants to comply with the Fair Housing and Equal Credit Opportunity Acts

Project steps:

- ➔ ETL
- ➔ ML
- ➔ Store on SQL
- ➔ HTML/CSS
- ➔ Flask app
- ➔ Web hosting

Dashboard

ETL

H_o = coef is 0

H_1 = coef is **not 0**

- If the p-value is less than 0.05, we reject the null hypothesis
- Features with a p-value less than 0.05 are significant to the model

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.327
Model:                            OLS   Adj. R-squared:                  0.306
Method:                 Least Squares   F-statistic:                     16.03
Date:                Thu, 16 Jun 2022   Prob (F-statistic):           1.94e-40
Time:                        22:03:36   Log-Likelihood:                -277.84
No. Observations:                 614   AIC:                             593.7
Df Residuals:                     595   BIC:                             677.7
Df Model:                          18
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                       0.1509      0.153      0.986      0.325      -0.150       0.452
ApplicantIncome          1.094e-06   3.29e-06      0.333      0.739   -5.36e-06    7.55e-06
CoapplicantIncome        -9.23e-06   5.73e-06     -1.610      0.108   -2.05e-05    2.03e-06
LoanAmount                 -0.0002      0.000     -1.005      0.315      -0.001       0.000
Loan_Amount_Term           -0.0002      0.000     -0.764      0.445      -0.001       0.000
Credit_History              0.7033      0.045     15.527      0.000       0.614       0.792
Gender_Female               0.0402      0.116      0.346      0.730      -0.188       0.269
Gender_Male                 0.0353      0.110      0.321      0.749      -0.181       0.252
Married_No                 -0.2120      0.252     -0.841      0.401      -0.707       0.283
Married_Yes                -0.1178      0.251     -0.469      0.639      -0.611       0.376
Dependents_0                0.0142      0.115      0.123      0.902      -0.211       0.240
Dependents_1               -0.0527      0.119     -0.442      0.658      -0.287       0.181
Dependents_2                0.0455      0.119      0.381      0.703      -0.189       0.280
Dependents_3+               0.0104      0.126      0.083      0.934      -0.237       0.257
Education_Graduate          0.1040      0.078      1.325      0.186      -0.050       0.258
Education_Not Graduate      0.0469      0.079      0.589      0.556      -0.109       0.203
Self_Employed_No            0.0139      0.071      0.195      0.846      -0.126       0.154
Self_Employed_Yes           0.0170      0.081      0.210      0.834      -0.143       0.177
Property_Area_Rural        -0.0060      0.057     -0.106      0.916      -0.118       0.106
Property_Area_Semiurban     0.1225      0.055      2.220      0.027       0.014       0.231
Property_Area_Urban         0.0344      0.055      0.625      0.533      -0.074       0.143
==============================================================================
Omnibus:                       91.264   Durbin-Watson:                   1.944
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              130.524
Skew:                          -1.112   Prob(JB):                     4.54e-29
Kurtosis:                       3.397   Cond. No.                     5.95e+19
```

- The different MS algorithms underperformed
    - Each MS algorithm either produced overfitting or underfitting

```
Model: LinearRegression
Train score: 0.3450858320862960б
Test Score: 0.18400955120272888

Model: KNeighborsRegressor
Train score: 0.3670644597423336
Test Score: 0.006243452755080536

Model: RandomForestRegressor
Train score: 0.8963577979671551
Test Score: 0.18500993086109374

Model: ExtraTreesRegressor
Train score: 1.0
Test Score: -0.07290861093651779

Model: AdaBoostRegressor
Train score: 0.349653915530037
Test Score: 0.1767992812257505

Model: SVR
Train score: 0.4055056910364776
Test Score: 0.2103514652214652
```

- To get above the 77% accuracy requirement, we decided to try other classification models

- The final model we utilized was the XGB classifier as it reduces the residuals

```
expected_y  = y_test
y_pred = model.predict(X_test)
print(metrics.classification_report(expected_y, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.51      0.62        49
           1       0.80      0.93      0.86       105

    accuracy                           0.80       154
   macro avg       0.79      0.72      0.74       154
weighted avg       0.80      0.80      0.79       154
```

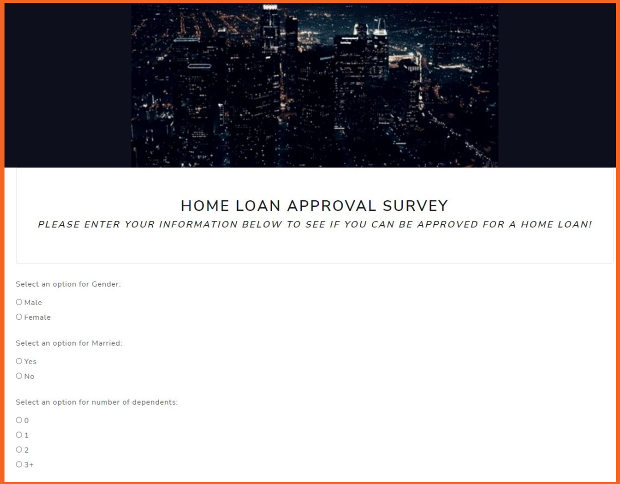
SQL Storage

- ➔ We used Postgres to store our transformed data in a relational database.
- ➔ We did that so that when the data is updated/more entries are added to further train the model, the new data is easily incorporated.
- ➔ Data movement is expensive.

HTML/CSS Process

- Code was surprisingly straightforward
- Use of “Form” functions consisting of single-option and user-input cells for a user to make their choices and input their own data
- The only CSS that was used was for a custom template that is used for the site itself

```html
<div class="container">
    <div class="jumbotron text-center bg-light">
        <h1>Home Loan Approval Survey</h1>
        <h4><i>Please enter your information below to see if you can be approved for a home loan!</i></h4>
    </div>
</div>
<div class="container">
    <p>Select an option for Gender:</p>
    <form method = "post" id = "myform">
        <input type="radio" id="male" name="gender" value="Male">
        <label for="male">Male</label><br>
        <input type="radio" id="female" name="gender" value="Female">
        <label for="male">Female</label><br>
    <br>
    <p>Select an option for Married:</p>
        <input type="radio" id="yes" name="married" value="Yes">
        <label for="yes">Yes</label><br>
        <input type="radio" id="no" name="married" value="No">
        <label for="no">No</label><br>
    <br>
    <p>Select an option for number of dependents:</p>
    <input type="radio" id="0" name="dependents" value="0">
    <label for="0">0</label><br>
    <input type="radio" id="1" name="dependents" value="1">
    <label for="1">1</label><br>
    <input type="radio" id="2" name="dependents" value="2">
    <label for="2">2</label><br>
    <input type="radio" id="3+" name="dependents" value="3+">
    <label for="3+">3+</label><br>
    <br>
    <p>Please select educational level:</p>
        <input type="radio" id="graduate" name="education" value="Graduate">
        <label for="education">Graduate</label><br>
        <input type="radio" id="non-graduate" name="education" value="Non-Graduate">
        <label for="education">Non-Graduate</label><br>
        <br>
            <p>Please indicate whether you are self-employed:</p>
            <input type="radio" id="yes" name="employment" value="Yes">
            <label for="yes">Yes</label><br>
            <input type="radio" id="no" name="employment" value="No">
            <label for="no">No</label><br>
```



_ Flask

```python
from binascii import Incomplete
from flask import Flask, request, render_template
import pickle
import numpy as np

app = Flask(__name__)

@app.route('/')
def hello():
    return render_template('index.html')


@app.route('/', methods =["POST"])
def gfg():
    if request.method == "POST":
        # getting input with name = fname in HTML form
        gender = request.form.get("gender")
        print(gender)
        married = request.form.get("married")
```

```python
        data = []
        data.append(int(income_applicant))
        data.append(int(income_coapplicant))
        data.append(int(loan))
        data.append(int(credit))
        if gender == "Male":
            data.append(0)
            data.append(1)
        else:
            data.append(1)
            data.append(0)
        if married == "Yes":
            data.append(0)
            data.append(1)
        else:
            data.append(1)
            data.append(0)
        if dependents == "0":
```

Important Notes:

- We used Flask to develop our web app
- Got all the user inputs and printed them
- Next step we appended the inputs to an empty list called data
- Finally we used pickle to load the ML model saved in a sav format
    - Pickle is great for serialization
    - Very quick → fast execution time
- Used the model to predict the result which was either Loan Approved or Loan Not Approved

```python
        print(data)
        loaded_model = pickle.load(open("finalized_model.sav", 'rb'))
        result = loaded_model.predict(np.array(data, dtype=np.int32).reshape(1,-1))
        print(f"result = {result}")
        if result[0] == 0:
            output = "Not Approved"
        else:
            output = "Approved"
```

Website run through

Future considerations

Questions?