Team Presentation Report

Info 7390

Advanced Data Science

Default credit card client Analysis

Presented By:

Birwa Galia

(001824469)

galia.b@husky.neu.edu

Milony Mehta

(001869360)

mehta.mil@husky.neu.edu

Shantanu Deosthale

(001851612)

deosthale.s@husky.neu.edu

This report summarizes the analysis of the clients which would be able to pay credit card which would be generated next month based on the data present of previous month bills, person's personal details and credit card limit.

We are getting the data from:

https://s3.amazonaws.com/assignment3datasets/default+of+credit+card+clients.xls

1) **Data Wrangling**
   - Exploratory Data Analysis on Python.
   - Manipulating the data necessary

2) **Dockerizing the process:**
   - Created docker image to pickle the various models and get the accuracy metrics
   - Airflow pipeline

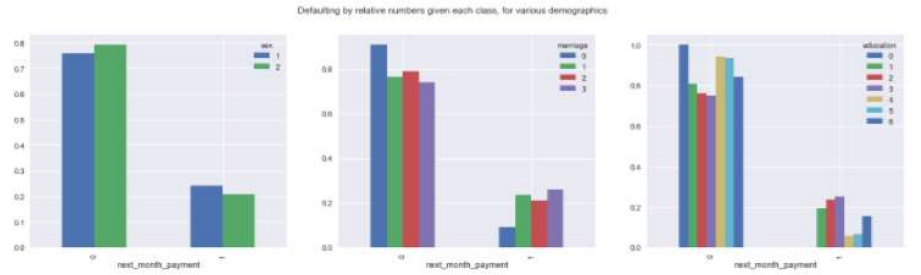
3) **Building and Evaluating models:**
   - Classification using Random forest, Logistic Regression, KNN Classifier, Extra Trees Classifier, Bernoulli Navies Bayesian

4) **Creating User interface for the Bank**
   - **Bank:** The bank will be able to predict whether user will be able to pay the next month bill or not.
   - **The output will be either 0 or 1 where 0 is NO and 1 is Yes**

PART 1: DATA WRANGLING

1. Exploratory Data Analysis

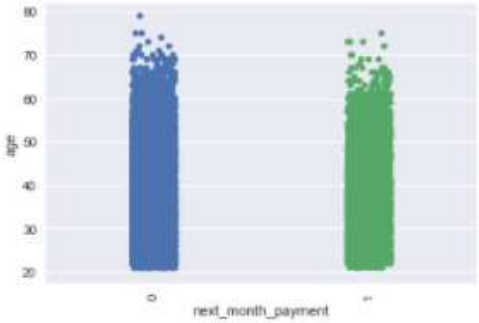
```
fig, ax = plt.subplots(1,3)
fig.set_size_inches(20,5)
fig.suptitle('Defaulting by relative numbers given each class, for various demographics')
d = dataset.groupby(['next_month_payment', 'education']).size().unstack(level=1)
d = d / d.sum()
p = d.plot(kind='bar', ax=ax[2])
d = dataset.groupby(['next_month_payment', 'marriage']).size().unstack(level=1)
d = d / d.sum()
p = d.plot(kind='bar', ax=ax[1])
d = dataset.groupby(['next_month_payment', 'sex']).size().unstack(level=1)
d = d / d.sum()
p = d.plot(kind='bar', ax=ax[0])
```




```
## Plotting Strip plot for the categorical value Day of the Week
```

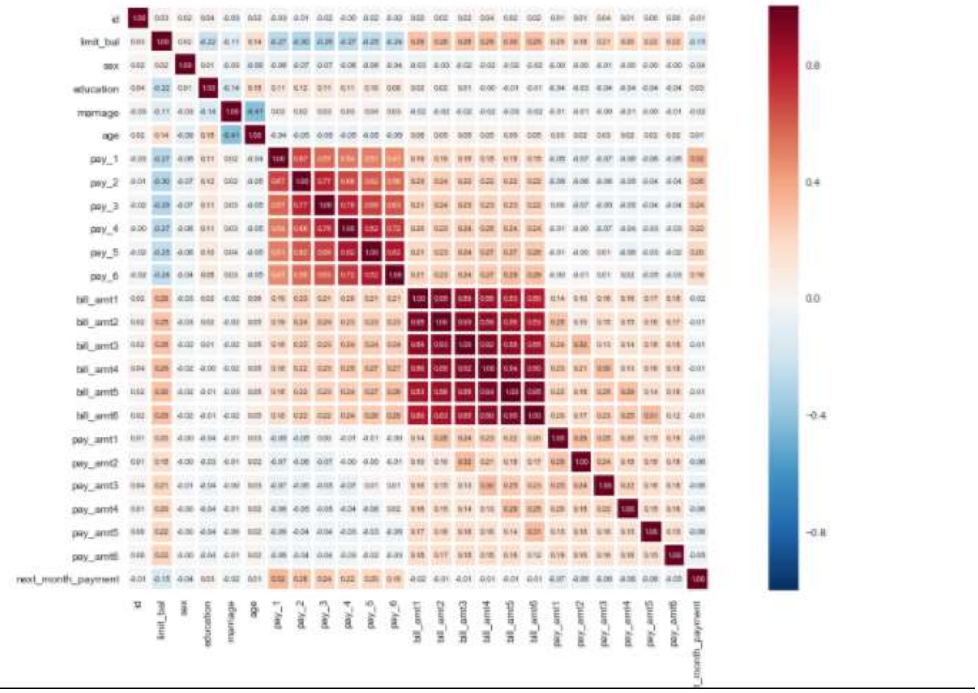
- The first graph shows that the no. of female that cannot pay the next_month _payment is more than that of male.
- The second graph shows that the single people wont be able to pay next month payment as compared to the others
- The third graph shows the person who has completed high school will be able to pay.

```
In [46]: ## Plotting Strip plot for the categorical value Day of the Week
         g = sns.stripplot(x='next_month_payment', y='age', data=dataset, jitter=True)
         plt.xticks(rotation=90)
Out[46]: (array([0, 1]), <a list of 2 Text xticklabel objects>)
```



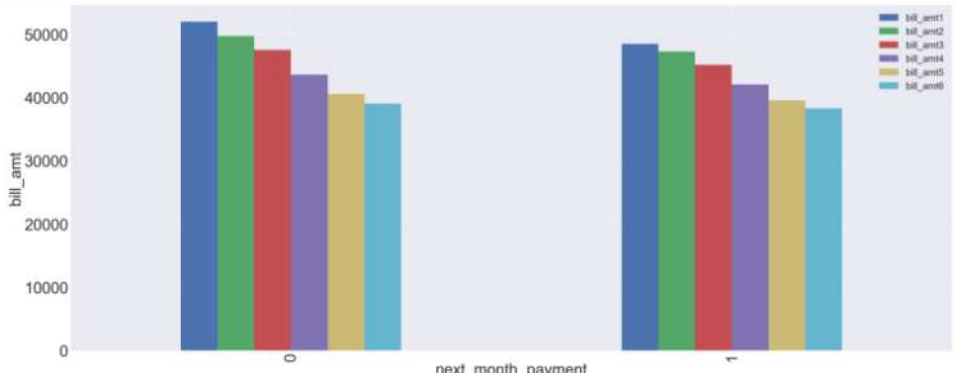
This graph that there is high possibility of the people whose age is from 20 – 50 will be able to pay.

In [48]:
```
## Checking if relation is significant
plt.figure(figsize=(20,10))
sns.heatmap(corr, cbar = True, square = True, annot=True, linewidths = 2, fmt='.2f',annot_kws={'size':7})
#sns.plt.title('Heatmap of Correlation Matrix')
plt.show()
```



In the above correlation, the pay_n columns is highly correlated to target value while limit_bal is correlated to bill_amt and pay_amt

In [54]:
```
bill_amt = dataset.groupby('next_month_payment')['bill_amt1' , 'bill_amt2' , 'bill_amt3', 'bill_amt4' , 'bill_amt5' , 'bill_amt6']
bill_amt.plot(kind = 'bar' , figsize = (50,20) , fontsize = 50)
plt.xlabel('next_month_payment', fontsize = 50)
plt.legend(fontsize = 30)
plt.ylabel('bill_amt' , fontsize = 50)
plt.show()
```



This graph shows that bill_amt is dependent on each other

Feature Engineering

```
In [123]: def feature_engineering(dataset):
              dataset = replacing_missing_values(dataset)
              filedu = (dataset.education == 5)|(dataset.education == 6)|(dataset.education == 0)
              dataset.loc[filedu,'education'] = 4
              filmarra = (dataset.marriage == 0)
              dataset.loc[filmarra,'marriage'] = 3
              fil = (dataset.pay_1 == -2) | (dataset.pay_1 == -1) | (dataset.pay_1 == 0)
              dataset.loc[fil, 'pay_1'] = 0
              fil = (dataset.pay_2 == -2) | (dataset.pay_2 == -1) | (dataset.pay_2 == 0)
              dataset.loc[fil, 'pay_2'] = 0
              fil = (dataset.pay_3 == -2) | (dataset.pay_3 == -1) | (dataset.pay_3 == 0)
              dataset.loc[fil, 'pay_3'] = 0
              fil = (dataset.pay_4 == -2) | (dataset.pay_4 == -1) | (dataset.pay_4 == 0)
              dataset.loc[fil, 'pay_4'] = 0
              fil = (dataset.pay_5 == -2) | (dataset.pay_5 == -1) | (dataset.pay_5 == 0)
              dataset.loc[fil, 'pay_5'] = 0
              fil = (dataset.pay_6 == -2) | (dataset.pay_6 == -1) | (dataset.pay_6 == 0)
              dataset.loc[fil, 'pay_6'] = 0
              dataset['AgeBin'] = pd.cut(dataset['age'], 6, labels = [1,2,3,4,5,6])
              dataset['AgeBin'] = pd.to_numeric(dataset['AgeBin'])
              return dataset
```

The value which is undefined have been manipulated

Model Performance

We tried to balance the data using SMOTE and NeverMiss, the sensitivity and specificity was getting better by SMOTE . We performed oversampling using SMOTE to get better accuracy.

Feature Selection:

1. Recursive Feature Elimination

	Model_Name	F1_score	Accuracy_score	True_Positive	False_Positive	True_Negative	False_Negative
0	ExtraTreesClassifier	0.840618	0.846672	4134	572	861	3779
1	RandomForestClassifier	0.813932	0.831372	4323	383	1193	3447
2	KNeighborsClassifier	0.747356	0.757222	3721	985	1284	3356
3	BernoulliNB	0.592905	0.679542	4170	536	2459	2181
4	LogisticRegression	0.617053	0.588166	2396	2310	1539	3101

2. Stepwise Selection

	Model_Name	F1_score	Accuracy_score	True_Positive	False_Positive	True_Negative	False_Negative
0	ExtraTreesClassifier	0.887073	0.892039	4374	332	677	3963
1	RandomForestClassifier	0.848055	0.857907	4312	394	934	3706
2	KNeighborsClassifier	0.769513	0.757650	3300	1406	859	3781
3	BernoulliNB	0.693754	0.740317	4170	536	1891	2749
4	LogisticRegression	0.664497	0.624973	2370	2336	1169	3471

Building and evaluating the models:

1. RandomForest Classifier

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix


rf = RandomForestClassifier(n_estimators=40, max_depth=10)

rf.fit(X_train, y_train)
prediction_train = rf.predict(X_test)

#r2_train = r2_score(y_train, prediction_train)
#mse_train = mean_squared_error(y_train, prediction_train)
#mae_train = mean_absolute_error(y_train, prediction_train)
#rmse_train = np.sqrt(mse_train)
#mape_train =  np.mean(np.abs((y_train-prediction_train) / y_train) ) * 100
x =confusion_matrix(y_test, prediction_train)
print(x)

print('Accuracy testing for Train data : {:.3f}'.format(accuracy_score(y_test, prediction_train)))
```

Similarly, for other models the X and Y has been fitted

```
In [127]: X = data[['id', 'limit_bal', 'sex', 'education', 'marriage', 'age', 'pay_1',
                    'pay_2', 'pay_3', 'pay_4', 'pay_5', 'pay_6', 'bill_amt1', 'bill_amt2',
                    'bill_amt3', 'bill_amt4', 'bill_amt5', 'bill_amt6', 'pay_amt1',
                    'pay_amt2', 'pay_amt3', 'pay_amt4', 'pay_amt5', 'pay_amt6']]

In [128]: y = data['next_month_payment']
```

This X are independent variables and Y is target variable

Pickling the models

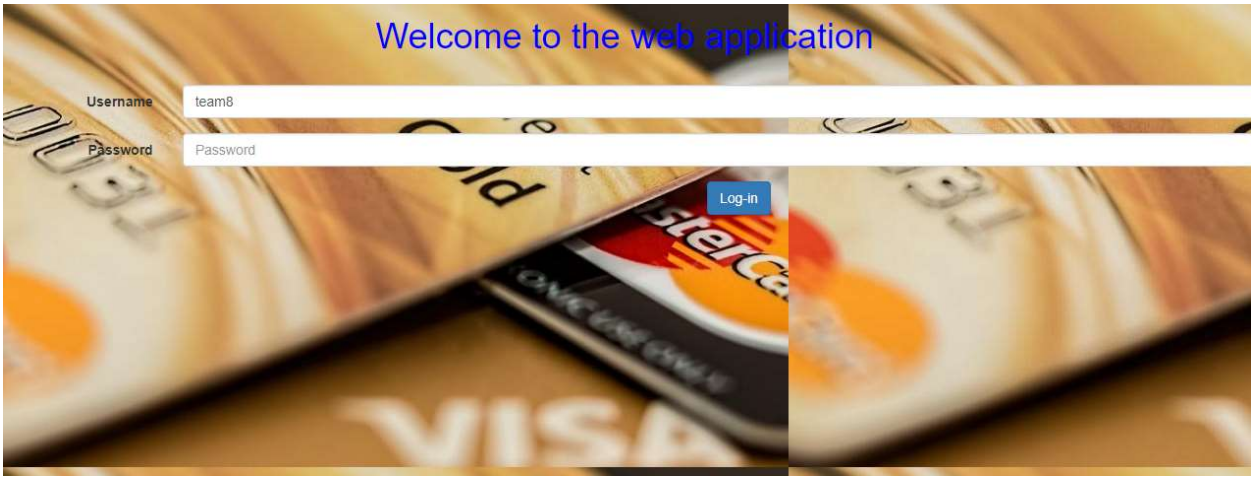
```
def k_n(dataset):
    x_train_res, x_val_res, y_train_res, y_val_res = train_test(dataset)
# instantiate learning model (k = 3)
    knn = KNeighborsClassifier(n_neighbors=4)

# fitting the model
    knn.fit(x_train_res, y_train_res)
    filename = 'knn_model.pckl'
    pickle.dump(knn, open(filename, 'wb'))

    # some time later...

    # load the model from disk
    K_nearest_model = pickle.load(open(filename, 'rb'))
    return K_nearest_model
```

This is the just the pickling of KNN model.

The app for determining the default next month prediction is developed in

- Python Flask
- HTML
- CSS (BootStrap)

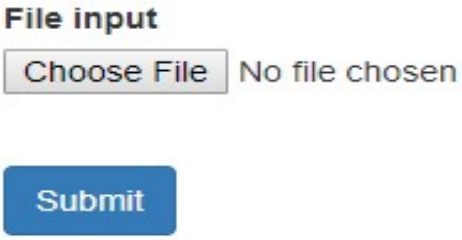
Web Application using Flask



This is our login page where user of the bank will put his/her credentials for login purpose. The input is static where Username = “team8” and password = “team8”.

This are the two ways user can input the data to analyze the result

1.) Uploads The CSV FIle

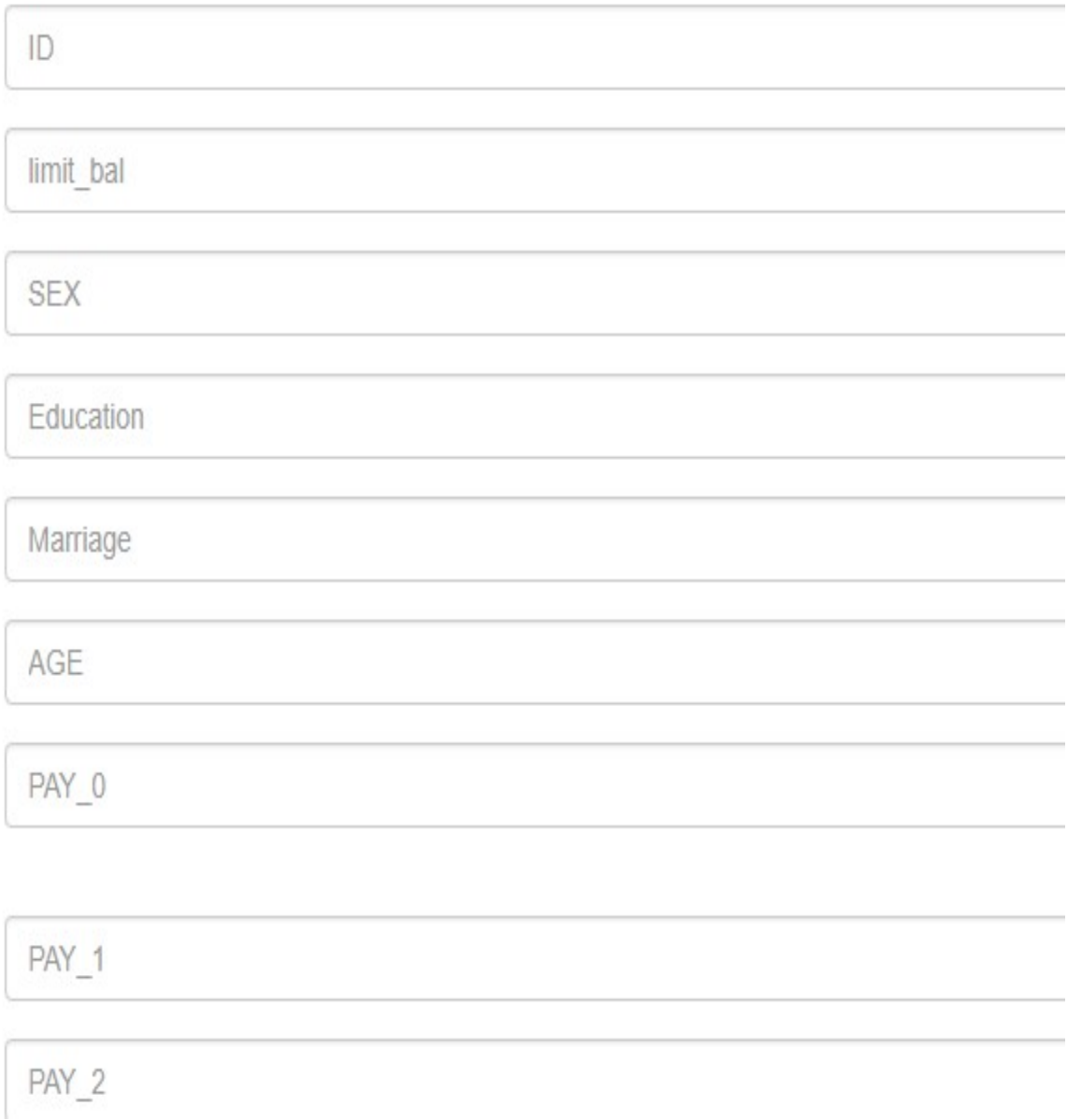



```
filename = secure_filename(file.filename)
dir_name = 'uploads/'
if not os.path.exists(dir_name):
    os.makedirs(dir_name)
file_path = os.path.join(dir_name, filename)
file.save(file_path)
try:

    output = upload_file_to_s3(file, cg.S3_BUCKET)
    print(output)
    dataset = pd.read_csv(file_path,header = 1)
    print(dataset)
    conn = S3Connection(cg.S3_KEY, cg.S3_SECRET_ACCESS_KEY)
    b = conn.get_bucket(cg.S3_BUCKET)
    for obj in b.get_all_keys():
        trial = obj.get_contents_to_filename(obj.key)
```

Takes files from the choose file and save it into local and uploads directed into the S3.

This is the page after user clicks upload csv till will only accept that format and downloads the pickle models from S3.

2.) Takes User Inputs In HTML Form

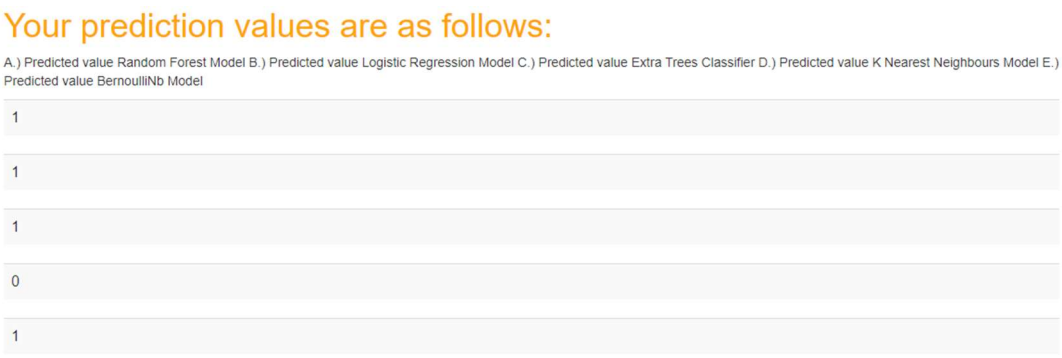


This is the page after user chooses fill form option where user puts the data manually

```
@app.route('/prediction_form/' , methods=["POST"])
def prediction_form():
    if request.method == "POST":
        conn = S3Connection(cg.S3_KEY, cg.S3_SECRET_ACCESS_KEY)
        b = conn.get_bucket(cg.S3_BUCKET)
        for obj in b.get_all_keys():
            trial = obj.get_contents_to_filename(obj.key)
        Id = request.form.get("id")
        limit = request.form.get("limit_bal")
        sex = request.form.get("sex")
        education = request.form.get("education")
        marriage =  request.form.get("marriage")
        age = request.form.get("age")
        pay0 = request.form.get("pay0")
        pay1 = request.form.get("pay1")
        pay2 = request.form.get("pay2")
        pay3 = request.form.get("pay3")
        pay4 = request.form.get("pay4")
        pay5 = request.form.get("pay5")
        pay6 = request.form.get("pay6")
        bill_amt1 = request.form.get("bill_amt1")
        bill_amt2 = request.form.get("bill_amt2")
        bill_amt3 = request.form.get("bill_amt3")
        bill_amt4 = request.form.get("bill_amt4")
        bill_amt5 = request.form.get("bill_amt5")
        bill_amt6 = request.form.get("bill_amt6")
        payment1 =  request.form.get("payment1")
        payment2 =  request.form.get("payment2")
        payment3 =  request.form.get("payment3")
        payment4 =  request.form.get("payment4")
        payment5 =  request.form.get("payment5")
```

```
        y = [prediction1 , prediction2, prediction3 , prediction4 , prediction5]
        Y = pd.DataFrame(y)
        y1 = pd.DataFrame(y[0], columns= {"Predicted value rf_model"})
        y2 = pd.DataFrame(y[1], columns= {"Predicted value lr_model"})
        y3 = pd.DataFrame(y[2], columns= {"Predicted value extra_tree_model"})
        y4 = pd.DataFrame(y[3], columns= {"Predicted value knn_model"})
        y5 = pd.DataFrame(y[4], columns= {"Predicted value bnb_model"})
        csv = y1.merge(y2, left_index = True, right_index = True , how= 'inner')
        csv = csv.merge(y3, left_index = True, right_index = True , how= 'inner')
        csv = csv.merge(y4, left_index = True, right_index = True , how= 'inner')
        csv = csv.merge(y5, left_index = True, right_index = True , how= 'inner')
        csv.to_csv(str(os.getcwd()) + "/Prediction Matrix.csv")
        return render_template("success.html", y = y)
        #a = map(float,a)
```

Takes user input values and performs prediction models by fetching Pickle models from S3. Save the prediction into CSV and saves it to the local directory.



After user clicks on the submit button it gets the predicted value in 0 and 1 format where 0 is user won't be able to pay for next month and 1 would be he will be able to pay