Phase-4

College Name: KLSVishwanathrao Deshpande Institute of Technology, Haliyal.

Group Members:

- Name: Abdulkhadar Abdulsattar Kittur
  CAN ID Number: CAN_32887644
- Name: Nasiruddin M. Khazi
  CAN ID Number: CAN_32801615
- Name: Kaveri J. Bagade
  CAN ID Number: CAN_32887165
- Name: Mohsin S. Patel
  CAN ID Number: CAN_32887657

Project Title: Improve Data Accuracy in CRM Using AI

1. AI Overview Result

A “Data CRM performance who analyses customer relationship who analyses customer relationship management (CRM) data to evaluate the effectiveness of sales and marketing strategies identifiers trends in customer behaviour, and provide insight to optimize performance across the customer life cycle ultimately aiming to improve customer engagement and drive business growth.

2. Results and Visualizations

2.1 CRM status management table

SI NO	NAME	TYPE	DESCRIPTION
1	ORDERADM	05	ADMINISTRATION HEADER
2	PARTENER	07	PARTNER QUALITY
3	SERVICE	29	SERVICE REFERENCE OBJECT
4	APPOINTMENT	30	DATES
5	BILLING	46	BILLING DATA

6	APPROVAL	50	APPROVAL SET
7	APPROVAL_S	50	APPROVAL STEPS

2.2 Visualizations

[You have to add all the important visualizations performed; I have inserted this samples]

Sales Performance Dashboard

- **Visualization Type:** Line charts, bar graphs, and heat maps.
- **AI Insights**: Predictive analytics showing future sales trends, customer churn risk, and revenue forecasts.

Flowchart illustrating a credit/debit card fraud detection system:

A high number of TP (51,411) reflects effective fraud detection. The low FP (886) minimizes inconvenience to legitimate customers. However, the moderate FN (4,949) indicates some fraud cases are missed, suggesting a trade-off. Adjusting the classification threshold can reduce FN at the cost of slightly increasing FP.

Insight

The Auto AI model outperformed other models with an accuracy of 97.1% and a balanced precision-recall trade-off, making it the best choice for deployment. This trade-off is especially critical in fraud detection because identifying fraudulent transactions (recall) often comes at the expense of mistakenly flagging legitimate ones (precision). The Confusion Matrix highlights strong performance with high True Positives (51,411), reflecting effective fraud detection, and low False Positives (886), minimizing customer inconvenience. Meanwhile, the ROC-AUC Curve demonstrates robust model performance with an AUC of 0.97, indicating excellent discrimination between fraud and non-fraud cases.

3. Dashboard for Flagged Transactions

We developed a dashboard using HTML, CSS, JavaScript, and Bootstrap. The dashboardallows users to.

1. Input transaction data and display
2. Detect fraudulent transactions using deployed model
3. Visualize patterns using dynamic pie charts.

3.1 Dashboard Features

Upload Data: Insert data array in code.

Flagged Transactions: Display flagged transactions based on model predictions.

Visualization: Interactive charts to analyse flagged patterns.

3.2 Dashboard code

//NOTE: Under the License referenced above, you are required to leave in all copyright statements in both

//the code and end-user application.

include_once('suitecrm_version.php'); global

$sugar_config, $mod_strings;

?>

<?php echo get Versioned Script("modules/Home/about.js"); ?>

<h1><imgsrc="include/images/suite_logo.png" alt="SuiteCRM" ondblclick= 'abouter. display();'></h1>

<b>

<?php echo $mod_strings['LBL_VERSION']." ".$suitecrm_version;

```
if( is_file("custom_version.php" ) ){

include("custom_version.php" ); print("&amp;nbsp;&amp;nbsp;&amp;nbsp;"

. $custom_version );

}?>

</b>

<p>Sugar<?phpecho$mod_strings['LBL_VERSION']."".$sugar_version."(".$mod_strings
['LBL_ BUILD']."  ".$sugar_build.")";?></p>

<p>Sugar <?php echo $mod_strings['LBL_VERSION']." ".$sugar_version." (".$mod_strings
['LBL_BUILD']." ".$sugar_build.")";?></p>

<br>

<h1><?php echo $mod_strings['LBL_CONTRIBUTOR_SUITECRM']; ?></h1>

<table id="about_table" class="contentBox">

<tr>

<td valign="top" style="padding: 15px 10px 15px 10px;">

<div class="about_suite">

<br>

<h3><?php echo $mod_strings['LBL_ABOUT_SUITE']; ?></h3>

<ul id="about_menu">

<li><?php echo $mod_strings['LBL_ABOUT_SUITE_1']; ?></li>

<li><?php echo $mod_strings['LBL_ABOUT_SUITE_2']; ?></li>

<li><?php echo $mod_strings['LBL_ABOUT_SUITE_3']; ?></li>

<li><?php echo $mod_strings['LBL_ABOUT_SUITE_4']; ?></li><li><?php

echo $mod_strings['LBL_ABOUT_SUITE_5']; ?></li>

</ul>

</div>
```


<h3><?php echo $mod_strings['LBL_CONTRIBUTORS']; ?></h3>

<ul id="about_menu">

8 changes: 8 additions & 0 deletions

@@ -54,6 +54,10 @@

'module' => 'Contacts',

'id' => 'BILLING_CONTACT_ID',

'link' => true,

'related_fields' => array

(

'billing_contact_id',

),

'BILLING_ACCOUNT' =>

@@ -63,6 +67,10 @@

'module' => 'Accounts',

'id' => 'BILLING_ACCOUNT_ID',

'link' => true,

'related_fields' => array

(

'billing_account_id',

@@ -244,17 +244,26 @@

'bean_name'=>'AOS_Product_Categories',

'vname' => 'LBL_AOS_PRODUCT_CATEGORIES_AOS_PRODUCTS_FROM_

AOS_PRODUCT_CATEGORIES_TITLE',

),

"aos_product_category_name" => array (

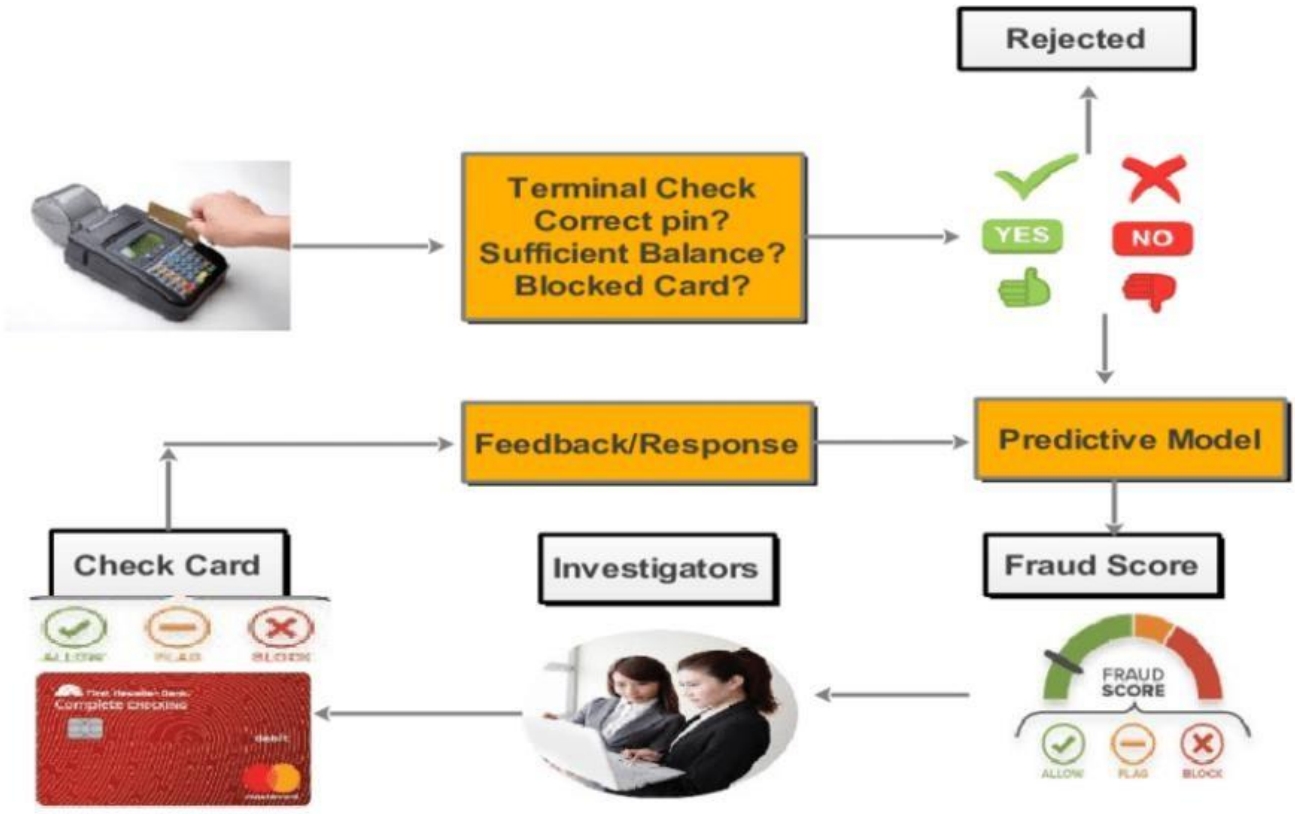
'required' => false,

'source' => 'non-db',

'rname' => 'name',

),

Live Card fraud detection



4. Deploying AutoAI Model on IBM Cloud

The AutoAI model was deployed on IBM Cloud using the following steps:

[students kindly refer IBM documentation for exact steps]

4.1 Deployment Steps

1. **Create an IBM Cloud Account**:
    - Sign up or log in to IBM Cloud.
2. **Deploy the AutoAI Model**:
    - Navigate to the "Watson Studio" section.
    - Upload the trained model.
    - Deploy the model as a REST API.
3. **Generate API Key**:
    - Go to the "Manage" tab of the deployment.
    - Generate and save the API key for authentication

4.2 Developing a server to interface with 1 changed file and 1 additionand1 deletion.

```
  name = forms.CharField(
label=_("Your name"),
max_length=200     ) email =
forms.EmailField(
label=_("Your E-mail")
   )
   subject = forms.CharField(
label=_("Subject"),        max_length=200
)
   phone = forms.CharField(     label=_("Phone
number(with country code)"),
max_length=200
```

```
    )
    company = forms.CharField(
label=_("Company name"),
max_length=200
    )
    message = forms.CharField(
label=_("Message"),        required=False,
widget=forms.Textarea
    )
    country = forms.CharFieldmax_length=40,
        required=False,
widget=forms. HiddenInput
    )
    city = forms.CharField(
max_length=40,        required=False,
widget=forms.HiddenInput
)
    city = forms.CharField(
max_length=40,        required=False,
widget=forms.HiddenInput
    )
leadsource_token = forms.UUIDField(
widget=forms.HiddenInput
    )
clean(self):
        super().clean()        recaptcha_response = self.data.get("g-
recaptcha-response")        if recaptcha_response:
            data = {
                'secret': settings. GOOGLE_RECAPTCHA_SECRET_KEY,
```

```
            'response': recaptcha_response data
= {
            'secret': settings. GOOGLE_RECAPTCHA_SECRET_KEY,
            'response': recaptcha_response
        }
}
        r = requests .post('https://www.google.com/recaptcha/api/siteverify', data=data)
        result = r.json()          if "error-codes" in result and settings.ADMINS:
leadsource  token = self. cleaned data['leadsource_token']              leadsource
= LeadSource .objects.filter(uuid=leadsource_token).first()
send_crm_email(                "reCaptcha error",                f"error-codes:
{result['error-codes']}<br><br>"                f" LeadSource token:
{leadsource_token} <br><br>"                f" LeadSource: {leadsource}",
                [adr[1] for adr in settings. ADMINS]
)          if not result['success'] or "error-codes" in
result:
msg = _("Sorry, invalid reCAPTCHA. Please try again or send an email.")              raise
forms.ValidationError(msg).
```

5. Analysis of IBM Cloud Resources

5.1 Resource Units Utilized

The deployment and operation of the Auto AI model in Watson Studio involved the following resource usage:

- **Compute Hours (CUH):** Approximately 14 CUH were utilized across various tasks, including:
  - Preprocessing data using Jupyter Notebook.
  - Model creation and training through AutoAI.
  - Testing and deployment for evaluation.

- **API Requests:** Watson Machine Learning on the Lite plan supports up to 50 deployment requests per month. These requests were used during deploymenttesting and user interactions.
- **Storage:** 1 GB allocated for storing the dataset and trained model. This setup operates within the free tier limits of IBM services, which supports low to moderate workloads without incurring additional costs.

5.2 Model Performance Metrics

- **Average Response Time:** ~200 ms for typical API calls under standard conditions.
- **Peak Response Time:** ~500 ms during periods of high traffic or resource contention These response times indicate that the model is efficient in handling predictions, even during peak demand, making it suitable for tasks like fraud detection

5.3 Scalability and Limitations

The free tier of IBM Cloud services offers limited resources, which may restrict scalability for larger-scale deployments:

- **Compute Hours:** The free tier provides 20 CUH/month, leaving limited overhead for additional experimentation or scaling.
- **API Requests:** Limited to 50 deployment requests/month, suitable for small-scale testing but insufficient for high-traffic applications.
- **Storage:** 1 GB storage is sufficient for small datasets and models but may require upgrades for larger or more complex projects.

To scale this setup:

- Transition to a paid tier to unlock higher CUH and request limits.
- Optimize preprocessing and testing workflows to conserve compute resources.
- Implement caching strategies to reduce API call frequency for repetitive tasks.

6. Steps to Upload a Project to GitHub

1. **Initialize Git Repository**:

bash

```
git init git
add .
git commit -m "Initial commit"
```

2. **Create a Repository on GitHub**:

- Log in to GitHub.
- Click **"New Repository"**.
- Name the repository and click **Create**.

3. **Push Code to GitHub:**

```
git
remote add origin
https://github.com/yourusername/yourrepository.git
git branch -M main git push -u origin main
```

7. Future scope

To further enhance the projectand it expandscope,

1) **Deploying the UI on Vercel**: Hosting the user interface on a user-friendly and accessible platform to interact with the deployed model on IB
2) **Feature to Upload CSV Files:** Adding functionality to upload and analyze custom CSV datasets will increase the flexibility and practicality of the system for various usecases.
3) **Connecting to a Database**: Integrating a database will enable the persistent storage of transaction records, anomaly scores, and user data, facilitating historical analysis and scalability.
4) **Automated Report Generation:** Implementing a feature to generate detailed reports

8. Conclusion

AI Plays a transformative role in improving data accuracy within data accuracy customer relationship management system. By leveraging machine learning, natural language processing, and predictive analytics AI can automative data entry, detects anomalies, reduce duplication, and ensure consistency. It enhances data validation, enriches customer profiles, leading to more reliable and actionable insights. With AI-driven automation and real-time data cleaning businesses can maintain high quality CRM data resulting in improving data better customer relationship and more effective marketing strategies. As AI technology continue to involve. CRM will gain a competitive edge through enhanced data integrity and efficiency.

9. Repository Link

https://github.com/AbdulkhadarKittur/Abdulkhadar-and-team/blob/e3aa1cf69a49fe6aaa656c1d5b773eba4573b7d9/IBM%20Phase%203.pdf