Carleton University

5301A: APPLIED ANALYTICS FOR TECH MGT

Documentation for Assignment 2

Group 9

Submitted by:

Lamya Islam Raisa
Muhammad Haris
Mani Dilipkumar
Shivani Sharma

1.Business Understanding

a. How would investors benefit from a model that predicts startup success?

While investors generally possess the skills to analyze companies, the abundance of options makes it challenging for them to efficiently evaluate and choose among final options. Prediction models can assist by enabling informed decisions based on evidence beyond their understanding, aiding in narrowing down the selection.

1. Our model will help the investors in making the right decision.
2. It will reduce the dependency on the venture capitalist to be a financial wizard or business driven.
3. The catch is investing in this early-stage startups are a bit risky but with the help of this model. Investors can make a find good business opportunities.

b. Why does it make sense to consider the lack of outcomes recorded in the Crunchbase database to be a sign of failure?

The lack of outcomes has the potential to bias the data set, resulting in inaccurate predictions. If a company is not listed as acquired or still active (is_closed is True), it implies that the company may have ceased its operations. In the absence of positive outcomes, closure is a reasonable assumption.

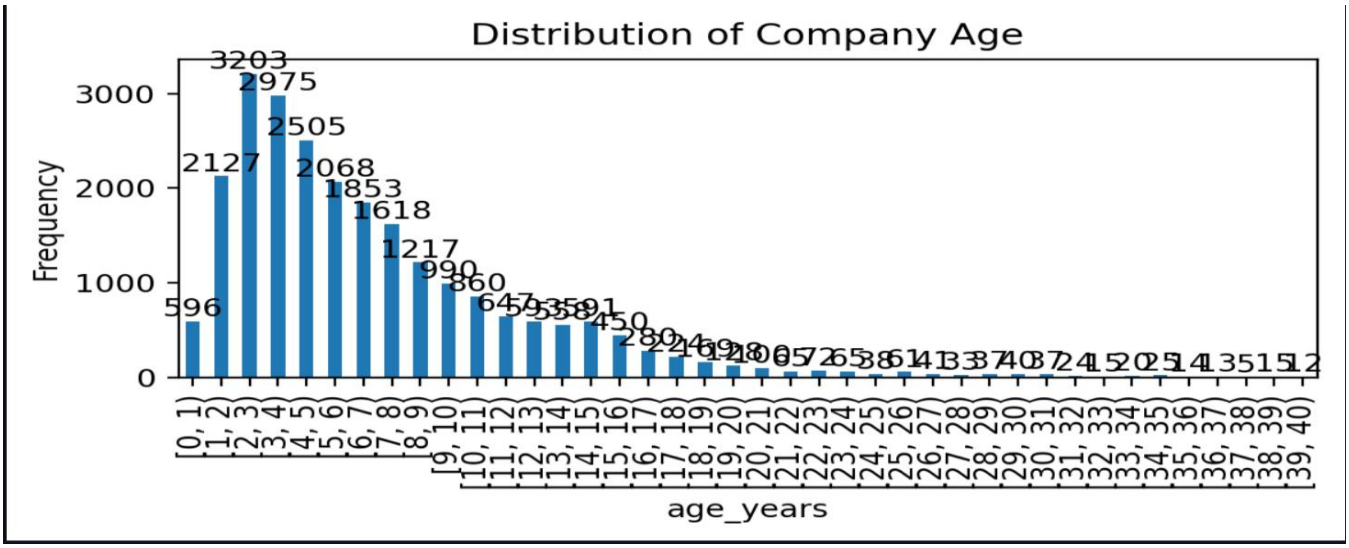
2. Data understanding and preparation

a. Show the distribution of age in years. Choose a suitable age range, so we can see each year in the plot. Which year is most frequent in the dataset?

Prompt used:

- Convert age from days to years.
- Eliminate missing values in the age column.
- Generate a bar chart for ages in the range 0 to 40.
- Display frequency labels vertically on the bars.
- Use a reduced font size for better aesthetics.
- Set the gap between columns to twice the size of the bar width.

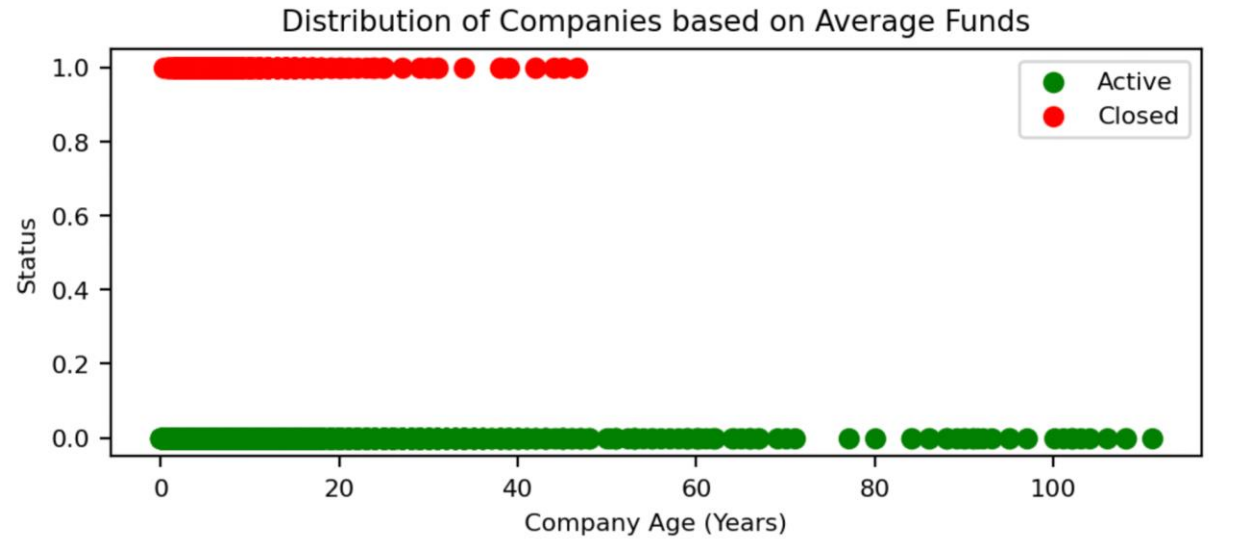
The following bar chart shows the distribution of age in years. The age range of 0-40 has been chosen to see each year in the plot. The most frequent year in the database is 3.



b. Create another visualization of the dataset that helps us better understand the nature of the data. Again, identify what prompt you used, and show the plot.

Prompt used: Generate a plot illustrating the distribution of companies based on their average funds, differentiating between closed and active companies. Each point should represent a company, with the x-axis indicating the company age in days and the y-axis indicating whether the company is closed (red) or still active (green). Convert the company age (days) to years.

A scatter plot is created where each point represents a company. The x-axis shows the age of the company in years, and the y-axis indicates whether the company is closed or still active, with different colors for differentiation.



c. Remove all companies from the dataset that are older than 7 years or younger than 3 years of age. How many companies are there in the remaining data?

Total Number of companies remaining in the dataset that are between 3 years and 7 years: 9261

d. Create a new target variable (success) with two values: 1 for success and 0 for failure.

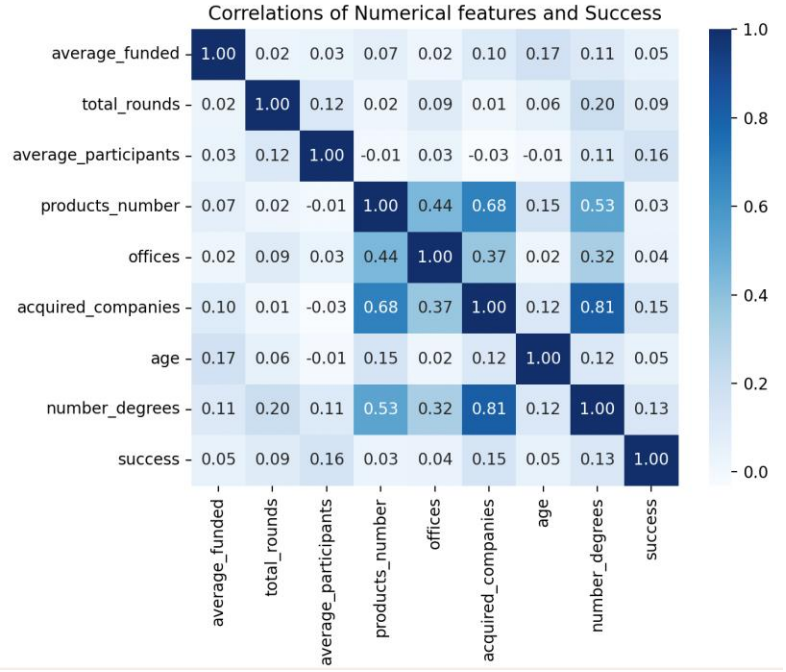
```
def success(row):
    if row['ipo']==True or row['is_acquired']==True and row['is_closed']==False:
        return 1
    elif row['ipo']== False and row['is_acquired']==False and row['is_closed']==False:
        return 0
    else:
        return 0
df['success']=df.apply(success, axis=1)
```

e. Combine the features related to the education levels of the founders (mba_degree, phd_degree, ms_degree, other_degree) into a new feature for the total number of degrees obtained by the founders (number_degrees).

```
# Combining Degrees Features
df['number_degrees'] = df[['mba_degree', 'phd_degree', 'ms_degree', 'other_degree']].sum(axis=1)
```

f. Identify the numerical features in the dataset (including the new feature created in e) and show their correlations with one another and the target in a heatmap. Are any features highly correlated with one another? What does this tell you?

The correlation matrix has been plotted against the following numerical features; ‘average_funded’,‘total_rounds’,‘average_participants’,‘products_number’,’offices’,’acquired_companies’,’age’,and ’number_degrees’.



As shown by the correlation matrix, there is no substantial numerical correlation between any feature and success in the dataset; nonetheless, the numerical feature ‘acquired_companies’ has the strongest correlation with success in the dataset. Average_participants come in second place, followed by total_rounds, age, and average_funded.

```
# Correlation between target variable and numerical features
df_numerical_features_and_success = pd.concat( objs: [df[numerical_features], df['success']], axis=1)
fig, ax = plt.subplots(figsize=(6.4, 4.8))
sns.heatmap(df_numerical_features_and_success.corr(), annot=True, fmt=".2f", ax=ax, cmap=plt.cm.Blues)
ax.set_title("Correlations of Numerical features and Success")
plt.show()
```

g. Identify the categorical features in the dataset. What will you need to do with the categorical features, before you can use them in the model? No code, just a description of what you need to do.

Categorical features like 'category_code,' 'country_code,' and 'state_code' need numerical conversion for machine learning. Methods like one-hot encoding or label encoding are used. Missing values can be filled with the most frequent category or advanced imputation techniques.

h. How do you avoid look-ahead bias? Can you use all the features in the dataset to predict startup success?

Look-ahead bias happens when future information influences predictions, leading to overoptimistic results. Attributes like 'ipo,' 'is_acquired,' and 'is_closed' were removed to create an unbiased model by preventing the use of unavailable future data in predictions.

```
# Dropping variables to avoid Look-ahead bias
df = df.drop( labels: 'ipo', axis=1)
df = df.drop( labels: 'is_acquired', axis=1)
df = df.drop( labels: 'is_closed', axis=1)
```

i. Some features are a large number of missing values. Which ones? Compute the missing values ratio for all features, ie the count of missing values (NA) in a column over the number of values.

‘company _id’, ‘category_code’, ‘average_funded’, ‘products_number’, ‘offices’, ‘acquired_companies’ and ‘age’ are some of the features that have missing values. The feature that had the largest number of missing values is ‘acquired_companies’, followed by ‘phd_degree’, ‘ms_degree’, etc.

	0
company_id	0
category_code	0.014
country_code	0
state_code	0
average_funded	0.1203
total_rounds	0
average_participants	0
products_number	0.8288
offices	0.0504
acquired_companies	0.9613
age	0
success	0
number_degrees	0

```
# Calculate the ratio of missing values in each column
1 usage
def missing_values_ratios(df):
    return df.isna().sum() / len(df)
print("Missing values ratios:")
print(missing_values_ratios(df))
```

j. Can we choose reasonable default values for the missing values or should any of these features be removed?

None of the features can be removed as they can potentially make an impact on the prediction hence a few considerable default values can be designated for missing values. We decided to designate 0 for the few missing values that we encountered on some attributes.

```
# Missing Values

df['mba_degree'] = df['mba_degree'].fillna(0)
df['phd_degree'] = df['phd_degree'].fillna(0)
df['ms_degree'] = df['ms_degree'].fillna(0)
df['other_degree'] = df['other_degree'].fillna(0)
```

3. Modeling

a) How do you choose training and test datasets from the full dataset?

Since the dataset is imbalanced, we have used stratified sampling to ensure that the distribution of classes is similar in both the training and test sets hence we considered 70% training and 30% test dataset.

```
# Split the data into 70% training and 30% testing, we use the stratified sampling in shuffling because our sample is not random

X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, shuffle=True, stratify=y, random_state=42)
```

b) Create a pipeline for pre-processing numerical and categorical features. This should also take care of missing values.

- For numerical features, we use a Pipeline that applies an imputer to replace all missing values and then a StandardScaler to standardize the features by removing the mean and scaling to unit variance.
- For categorical features, it applies a OneHotEncoder to convert categorical variables into a form that can be provided to ML algorithms to improve prediction.
- The ColumnTransformer applies these transformations to the appropriate columns in the dataset, ensuring all preprocessing steps are applied consistently across training and testing datasets.

```
def pre_processor(numerical_features, categorical_features):
    numerical_transformer = Pipeline(steps=[('imputer', SimpleImputer(strategy='constant', fill_value=0)),
                                            ('scaler', StandardScaler())])

    categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])

    preprocessor = ColumnTransformer(transformers=
                                     [('num', numerical_transformer, numerical_features),
                                      ('cat', categorical_transformer, categorical_features)])
    return preprocessor
```

c) Create two models: one using logistic regression, the other using random forests. As in the example discussed in class, allow the user to choose which model should be created and evaluated.

This pre_processor function preprocesses numerical and categorical features. Depending on the user's choice, a RandomForestClassifier or a LogisticRegression model is instantiated as the classifier.

These components are combined into a Pipeline, ensuring consistent application of preprocessing steps and the chosen classifier during model fitting and prediction.

```
preprocessor = pre_processor(numerical_features, categorical_features)
type_of_classifier = st.sidebar.radio("Select type of classifier", ("Random Forest", "Logistic Regression"))

if type_of_classifier == "Random Forest":
    classifier = RandomForestClassifier(random_state=1, max_depth=14, min_samples_split=2, min_samples_leaf=1,
                                        class_weight='balanced', )
elif type_of_classifier == "Logistic Regression":
    classifier = LogisticRegression(max_iter=2000, penalty='l2',
                                    class_weight='balanced')
model = Pipeline(steps=[('preprocessor', preprocessor), ('classifier', classifier)])
```

4. Evaluation and Interpretation

a) Which performance metrics are most suitable for evaluating your model to predict startup success? It helps to think about what performance metrics would matter most to an investor. Hint: It helps to read the article carefully.

- A model with high precision would provide just that a strong assurance that when it predicts a startup as successful, it is highly likely to be correct. **This precision is crucial as it can help mitigate investment risks.**
- The investor can thus focus on these high-probability successes, potentially leading to more effective investment strategies and outcomes.

b) The dataset is imbalanced. There are many more failed startups than successful startups. Suggest two ways to deal with imbalanced data and implement one of them.

There are several methods available to handle imbalanced data, including Stratified-K Fold and the Random Forest Classifier. In this case, we have implemented the Stratified-K Fold method.

c) Use stratified k-fold cross-validation to evaluate the model. Report the scores for each fold, and the averages.

Random Forest:

Scores for each fold (only positive class):

	0	1	2	3	4	mean
fit_time	1.5933	1.5721	1.6737	1.6226	1.6443	1.6212
score_time	0.1001	0.0886	0.0973	0.0745	0.0842	0.089
test_accuracy	0.8883	0.8815	0.8891	0.883	0.8829	0.8849
test_roc_auc	0.8447	0.8262	0.8405	0.825	0.8405	0.8354
test_precision	0.3724	0.3614	0.3696	0.3451	0.3469	0.3591
test_recall	0.4909	0.5455	0.4636	0.4455	0.4679	0.4827
test_f1	0.4235	0.4348	0.4113	0.3889	0.3984	0.4114

Logistic Regression:

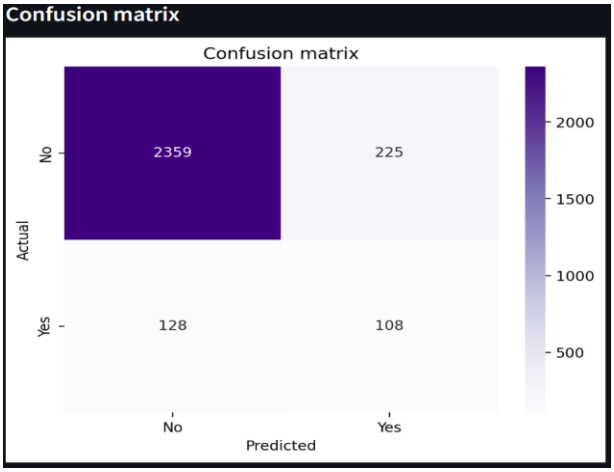
Scores for each fold (only positive class):

	0	1	2	3	4	mean
fit_time	0.1506	0.1668	0.1336	0.1336	0.147	0.1463
score_time	0.0162	0.0482	0.0162	0.0162	0.0326	0.0259
test_accuracy	0.7409	0.7485	0.7249	0.7424	0.724	0.7361
test_roc_auc	0.7756	0.7811	0.7482	0.7895	0.77	0.7729
test_precision	0.1887	0.2145	0.1802	0.201	0.1872	0.1943
test_recall	0.6364	0.7545	0.6455	0.7	0.6972	0.6867
test_f1	0.2911	0.334	0.2817	0.3124	0.2951	0.3029

d) For comparison, evaluate the model against the test dataset. Show the performance metrics in a heatmap.

The performance metrics for both the models are shown below:

Random Forest



Logistic Regression