Applied Statistics & Machine Learning

B9BA102

Continuous Assessment Two

Simran Shruthi [20023322]

Sunil Rao Shankar Rao[20030491]

Kaggle Dataset link -

https://www.kaggle.com/datasets/rabieelkharoua/predict-pet-adoption-status-dataset/data

Note - *Before we start the report, let's create a hypothetical company, background, and business task. This will help us show a clear and organized analysis using Statistical Regression. As business analysts, this approach will make sure our report presentation is easy to understand and relevant***.**

🐾 Paws & Claws Adoption Center 🐾

Business Requirement for Paws & Claws Adoption Center

Company Overview: Paws & Claws Adoption Center is dedicated to finding loving homes for abandoned and surrendered pets. The center takes in various animals, providing them with medical care, shelter, and rehabilitation until they are adopted. To improve its operations and enhance the adoption rates, Paws & Claws is seeking to leverage data-driven strategies.

Hypothetical Scenario: Paws & Claws Adoption Center has recently noticed a decline in adoption rates and an increase in the average time pets spend in the shelter. To address this, they have provided a dataset containing information about the pets and their adoption statuses. The center hopes that by analyzing this data, we can develop a predictive model to better understand the factors influencing adoption and implement strategies to improve adoption outcomes.

Model Objective: The objective is to develop a predictive model that accurately estimates the likelihood of each pet being adopted. By identifying the factors that significantly influence

adoption, Paws & Claws can implement targeted strategies to increase adoption rates, reduce shelter times, and improve overall animal welfare.

Critical analysis of Paws & Claws Adoption Center:

Q1. Data Preparation (What steps would you take to prepare your data? Discuss your approach)

Data importing and reading

Data preparation is the **processing of raw data** to create **accurate predictions**. In our dataset for the prediction of pet adoption status, we have created a machine learning model that can predict the same utilizing regression approaches such as **Linear Regression, Random Forest Regression, and Support Vector Machine Regression**. As part of the data preparation process, we cleaned, processed, and organized the raw data so that it may be analyzed and modeled. This technique is critical because our model's performance is highly tied to the quality of the data. To assure clean and meaningful data, we followed the methods below to achieve the best possible results.

- **Importing libraries :** We have imported necessary libraries such as pandas, sklearn, plotly. These libraries will be used for data handling, preprocessing and visualization as well.

```
from pandas import  read_csv, DataFrame,Series,get_dummies
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
from sklearn.linear_model import SGDRegressor
from plotly import graph_objs, figure_factory
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns
import pandas as pd
from sklearn.metrics import classification_report, roc_auc_score
```

- **ReadingData: read_csv():**ThisfunctionreadsthedatafromCinemaTickets file using the pandas function.
  Data exploration : Executing the commands gives us an overview of the dataset displaying column names,data types , null, non-null counts. The commands such as

  1. **data.info()** function provides information about the data, such as its attributes, non-null values, and the data types of each column.

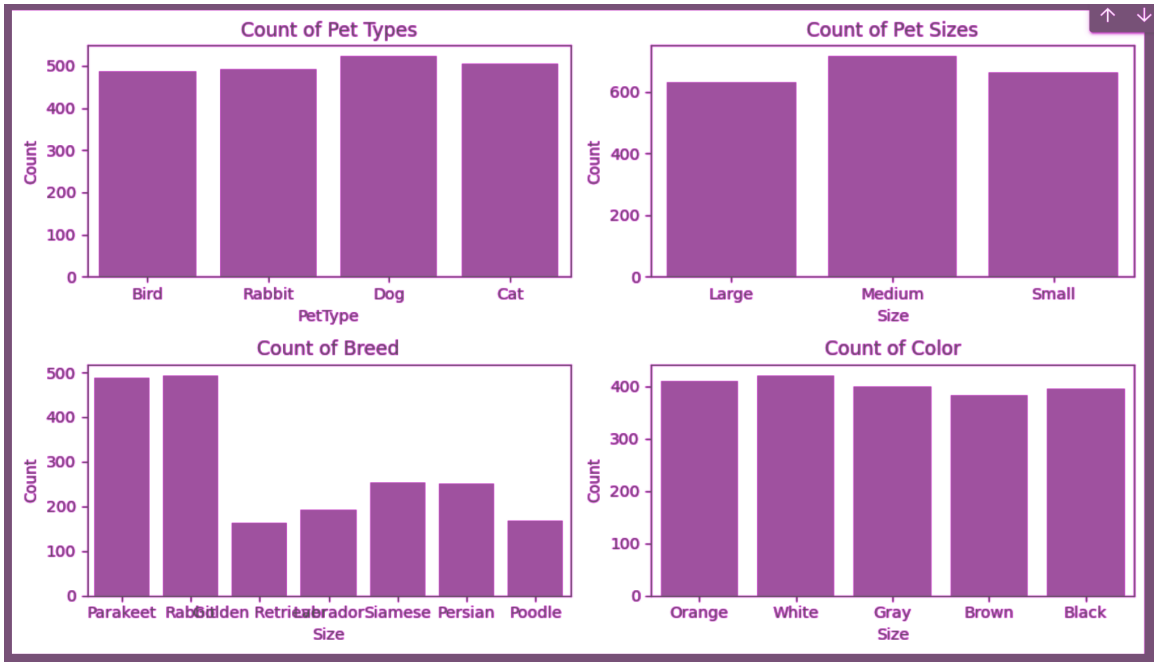
  2. **data.shape():** This function provides details about the dataset's number of rows and columns. The dataset's column and record counts are confirmed using the shape.

  3. **data.head()** - This function provides details about the dataset's rows.

  4. **data.describe()** This function gives the statistical description of the data. It provides statistical details about every attribute in the dataset, including the minimum, maximum, and means/average values.

We imported two libraries matplotlib for plotting and seaborn for creating bar plots.With the use of bar graphs, we were able to compare two categorical variables ('PetType' and 'Size') from a dataset (data1) side by side.

```
# Create a figure with 2 subplots side by side
fig, ((ax1, ax2),(ax3,ax4)) = plt.subplots(2,2,figsize=(10, 6))

# Bar plot for PetType
sns.countplot(x='PetType', data=data1, ax=ax1)
ax1.set_title('Count of Pet Types')
ax1.set_xlabel('PetType')
ax1.set_ylabel('Count')

# Bar plot for Size
sns.countplot(x='Size', data=data1, ax=ax2)
ax2.set_title('Count of Pet Sizes')
ax2.set_xlabel('Size')
ax2.set_ylabel('Count')

# Bar plot for Breed
sns.countplot(x='Breed', data=data1, ax=ax3)
ax3.set_title('Count of Breed')
ax3.set_xlabel('Size')
ax3.set_ylabel('Count')

# Bar plot for Color
sns.countplot(x='Color', data=data1, ax=ax4)
ax4.set_title('Count of Color')
ax4.set_xlabel('Size')
ax4.set_ylabel('Count')

# Show the plots
plt.tight_layout()
plt.show()
```



This code facilitates data analysis and interpretation by allowing you to compare and show the PetType,Size, Breed and Color category distributions in a single figure.

Data Encoding

We do Encoding to convert Categorical data into Numerical values. After reading data, we find out that there is need of encoding so we can convert all the columns that are with categorical type “object” into numerical data “integer”.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2007 entries, 0 to 2006
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   PetID               2007 non-null   int64  
 1   PetType             2007 non-null   object 
 2   Breed               2007 non-null   object 
 3   AgeMonths           2007 non-null   int64  
 4   Color               2007 non-null   object 
 5   Size                2007 non-null   object 
 6   WeightKg            2007 non-null   float64
 7   Vaccinated          2007 non-null   int64  
 8   HealthCondition     2007 non-null   int64  
 9   TimeInShelterDays   2007 non-null   int64  
 10  AdoptionFee         2007 non-null   int64  
 11  PreviousOwner       2007 non-null   int64  
 12  AdoptionLikelihood  2007 non-null   int64  
dtypes: float64(1), int64(8), object(4)
memory usage: 204.0+ KB
None
```

For encoding we use here two methods;

Method-1 Mapping: Where one variable has only 2 options or values that can be ranked and we map it in Yes/No or 1 & 0. Here in this dataset we used .map for 'PetType' and 'Size'
*Method-2 One-Hot Encoding:*In python language we use Get_Dummies - when one variable has more than two categories and if we use mapping method where convert original variable in list of sub-categories and assign assumed numerical value to each sub-category which may create low effect on each category. We can easily resolve it by using get_dummies code in python that divide the original column into encoded columns. Here are 2 independent variables i.e. Breed and color that further divide into encoding columns by using get-dummies code. Now we have 2007 rows and 23 columns shown in the table below. Use Code data2.info() to verify that all the columns Dtype converted to "int64" except weightKg.

```
data1['PetType'] = data1['PetType'].map({'Bird':0,'Cat':1,'Dog':2,'Rabbit':3})
data1['Size'] = data1['Size'].map({'Small':0,'Medium':1,'Large':2})
data1 = get_dummies(data1,columns = ['Breed', 'Color'], dtype=int)
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2007 entries, 0 to 2006
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   PetID                   2007 non-null   int64  
 1   PetType                 2007 non-null   int64  
 2   AgeMonths               2007 non-null   int64  
 3   Size                    2007 non-null   int64  
 4   WeightKg                2007 non-null   float64
 5   Vaccinated              2007 non-null   int64  
 6   HealthCondition         2007 non-null   int64  
 7   TimeInShelterDays       2007 non-null   int64  
 8   AdoptionFee             2007 non-null   int64  
 9   PreviousOwner           2007 non-null   int64  
 10  AdoptionLikelihood      2007 non-null   int64  
 11  Breed_Golden Retriever  2007 non-null   int64  
 12  Breed_Labrador          2007 non-null   int64  
 13  Breed_Parakeet          2007 non-null   int64  
 14  Breed_Persian           2007 non-null   int64  
 15  Breed_Poodle            2007 non-null   int64  
 16  Breed_Rabbit            2007 non-null   int64  
 17  Breed_Siamese           2007 non-null   int64  
 18  Color_Black             2007 non-null   int64  
 19  Color_Brown             2007 non-null   int64  
 20  Color_Gray              2007 non-null   int64  
 21  Color_Orange            2007 non-null   int64  
 22  Color_White             2007 non-null   int64  
dtypes: float64(1), int64(22)
memory usage: 360.8 KB
```

Featuring (X) & Labeling (Y)

All of the independent variables that are utilized to identify patterns qualify as features. Labels, however, include dependent variables that the model is predicted via. The target variable in this instance is "AdoptionLikelihood." This stage serves solely to confirm that we are headed in the right direction and to ensure our model evaluation in the training set is accurate.

Data Scaling

a.We have performed data scaling on input feature X using StandardScaler() from sklearn.preprocessing library. This will transform the numerical attributes to a standardized format. This process will normalize the data, ensuring that features are on a consistent scale.

b. Standardization helps prevent certain attributes with larger scales from dominating our model's learning process, ensuring fair consideration for all features in the dataset when making predictions.

Thus, we have obtained our desired dataset by performing all the processes as mentioned above, we will now proceed to perform all the regression methods and analyze their respective performances on our model.

Data Splitting

We divided the data into a training set and a testing set at this stage. We take this action because testing the model on the same dataset that we used for training would be unfair. As a result, we divided the data into two sets: one for testing and one for training. I divided the data into a 75:25 ratio in my case. In this instance, the training set size of 75% is what we're implementing to address Pet adoption prediction issues and is adequate to get a satisfactory model outcome.

However, a test size of 25% is sufficient to reliably assess the model's performance.

Heat Map:

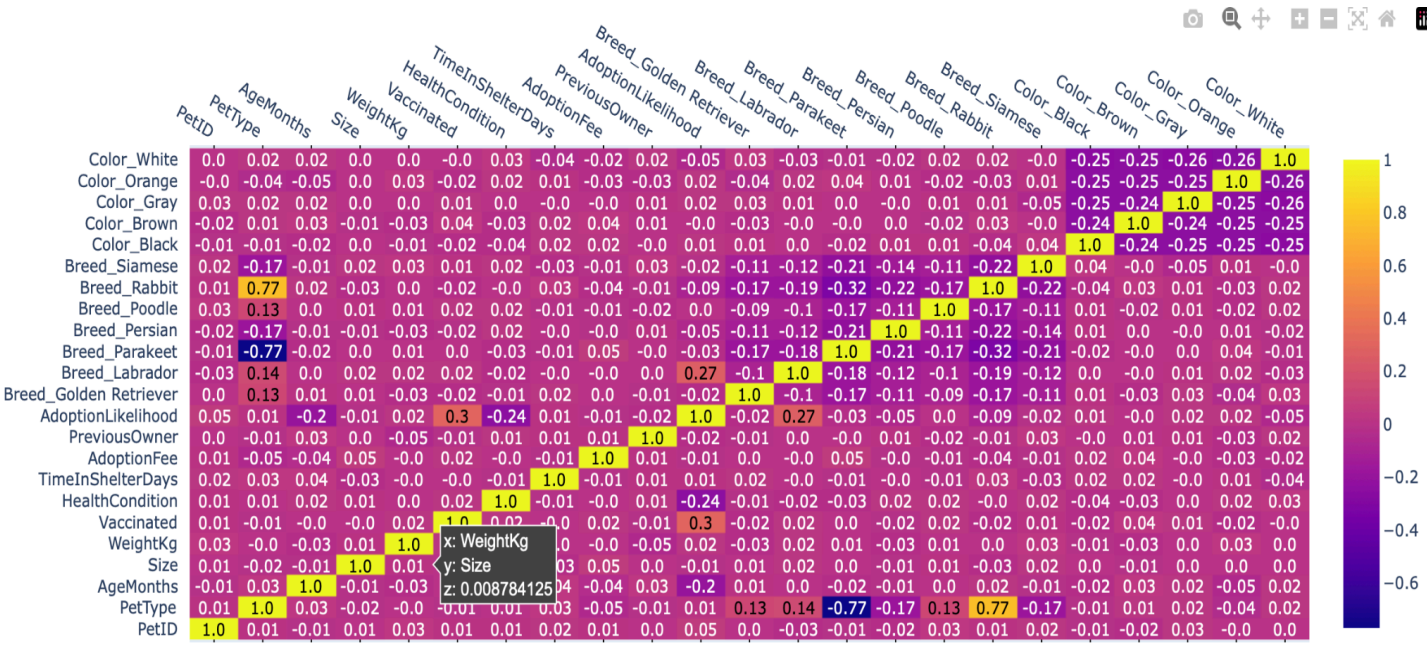
Heat Map is a graphical representation of dataset used for data visualization with structured matrix or rows and columns, where it visualizes volumes within a dataset. They are helpful in dealing with multicollinearity, defining correlation between multiple numerical variables, and spotting anomalies.

This segment includes the heat map depicting correlation between different features for our dataset.

1. **Importing libraries:** we have imported **figure_factory** from plotly library which offers a variety of functions to create tables, graphs etc . which helps in complex visualizations.

2. HeatMap creation:

- We plot the heatmap to identify the correlation between data.
- figure_factory.create_annotated_heatmap(): This function generates heatmap, all the annotations(numeric values) are added within each
  cell suitable for data interpretation.
- list(correlation.columns)labels the x-axis(columns).
- list(correlation.columns)labels the y-axis(rows).
- showscale=True: This enables the color scale, displaying the
  correlation strength using colors.
- heatmapData.show():This command will display generated heatmap
- After heatmap generation we came to know that 1 column is dependent on each other and hence we dropped the column **‘PetType’**.


PetID
PetType
AgeMonths
Size
WeightKg
Vaccinated
HealthCondition
TimeInShelterDays
AdoptionFee
PreviousOwner
AdoptionLikelihood
Breed_Golden Retriever
Breed_Labrador
Breed_Parakeet
Breed_Persian
Breed_Poodle
Breed_Rabbit
Breed_Siamese
Color_Black
Color_Brown
Color_Gray
Color_Orange
Color_White
Color_White 0.0 0.02 0.02 0.0 0.0 -0.0 0.03 -0.04 -0.02 0.02 -0.05 0.03 -0.03 -0.01 -0.02 0.02 0.02 -0.0 -0.25 -0.25 -0.26 -0.26 1.0
Color_Orange -0.0 -0.04 -0.05 0.0 0.03 -0.02 0.02 0.01 -0.03 -0.03 0.02 -0.04 0.02 0.04 0.01 -0.02 -0.03 0.01 -0.25 -0.25 -0.25 1.0 -0.26
Color_Gray 0.03 0.02 0.02 0.0 0.0 0.01 0.0 -0.0 -0.0 0.01 0.02 0.03 0.01 0.0 -0.0 0.01 0.01 -0.05 -0.25 -0.24 1.0 -0.25 -0.26
Color_Brown -0.02 0.01 0.03 -0.01 -0.03 0.04 -0.03 0.02 0.04 0.01 -0.0 -0.03 -0.0 -0.0 0.0 -0.02 0.03 -0.0 -0.24 1.0 -0.24 -0.25 -0.25
Color_Black -0.01 -0.01 -0.02 0.0 -0.01 -0.02 -0.04 0.02 0.02 -0.0 0.01 0.01 0.0 -0.02 0.01 0.01 -0.04 0.04 1.0 -0.24 -0.25 -0.25 -0.25
Breed_Siamese 0.02 -0.17 -0.01 0.02 0.03 0.01 0.02 -0.03 -0.01 0.03 -0.02 -0.11 -0.12 -0.21 -0.14 -0.11 -0.22 1.0 0.04 -0.0 -0.05 0.01 -0.0
Breed_Rabbit 0.01 0.77 0.02 -0.03 0.0 -0.02 -0.0 0.03 -0.04 -0.01 -0.09 -0.17 -0.19 -0.32 -0.22 -0.17 1.0 -0.22 -0.04 0.03 0.01 -0.03 0.02
Breed_Poodle 0.03 0.13 0.0 0.01 0.01 0.02 0.02 -0.01 -0.01 -0.02 0.0 -0.09 -0.1 -0.17 -0.11 1.0 -0.17 -0.11 0.01 -0.02 0.01 -0.02 0.02
Breed_Persian -0.02 -0.17 -0.01 -0.01 -0.03 -0.02 0.02 -0.0 -0.0 0.01 -0.05 -0.11 -0.12 -0.21 1.0 -0.11 -0.22 -0.14 0.01 0.0 -0.0 0.01 -0.02
Breed_Parakeet -0.01 -0.77 -0.02 0.0 0.01 0.0 -0.03 -0.01 0.05 -0.0 -0.03 -0.17 -0.18 1.0 -0.21 -0.17 -0.32 -0.21 -0.02 -0.0 0.0 0.04 -0.01
Breed_Labrador -0.03 0.14 0.0 0.02 0.02 0.02 -0.02 -0.0 -0.0 0.0 0.27 -0.1 1.0 -0.18 -0.12 -0.1 -0.19 -0.12 0.0 -0.0 0.01 0.02 -0.03
Breed_Golden Retriever 0.0 0.13 0.01 0.01 -0.03 -0.02 -0.01 0.02 0.0 -0.01 -0.02 1.0 -0.1 -0.17 -0.11 -0.09 -0.17 -0.11 0.01 -0.03 0.03 -0.04 0.03
AdoptionLikelihood 0.05 0.01 -0.2 -0.01 0.02 0.3 -0.24 0.01 -0.01 -0.02 1.0 -0.02 0.27 -0.03 -0.05 0.0 -0.09 -0.02 0.01 -0.0 0.02 0.02 -0.05
PreviousOwner 0.0 -0.01 0.03 0.0 -0.05 -0.01 0.01 0.01 0.01 1.0 -0.02 -0.01 0.0 -0.0 0.01 -0.02 -0.01 0.03 -0.0 0.01 0.01 -0.03 0.02
AdoptionFee 0.01 -0.05 -0.04 0.05 -0.0 0.02 -0.0 -0.01 1.0 0.01 -0.01 0.0 -0.0 0.05 -0.0 -0.01 -0.04 -0.01 0.02 0.04 -0.0 -0.03 -0.02
TimeInShelterDays 0.02 0.03 0.04 -0.03 -0.0 -0.0 -0.01 1.0 -0.01 0.01 0.01 0.02 -0.0 -0.01 -0.0 -0.01 0.03 -0.03 0.02 0.02 -0.0 0.01 -0.04
HealthCondition 0.01 0.01 0.02 0.01 0.0 0.02 1.0 -0.01 -0.0 0.01 -0.24 -0.01 -0.02 -0.03 0.02 0.02 -0.0 0.02 -0.04 -0.03 0.0 0.02 0.03
Vaccinated 0.01 -0.01 -0.0 -0.0 0.02 1.0 0.02 -0.0 0.02 -0.01 0.3 -0.02 0.02 0.0 -0.02 0.02 -0.02 0.01 -0.02 0.04 0.01 -0.02 -0.0
WeightKg 0.03 -0.0 -0.03 0.01 1.0 x: WeightKg -0.0 -0.05 0.02 -0.03 0.02 0.01 -0.03 0.01 0.0 0.03 -0.01 -0.03 0.0 0.03 0.0
Size 0.01 -0.02 -0.01 1.0 0.01 y: Size 0.05 0.0 -0.01 0.01 0.02 0.0 -0.01 0.01 -0.03 0.02 0.0 -0.01 0.0 0.0 0.0
AgeMonths -0.01 0.03 1.0 -0.01 -0.03 z: 0.008784125 -0.04 0.03 -0.2 0.01 0.0 -0.02 -0.01 0.0 0.02 -0.01 -0.02 0.03 0.02 -0.05 0.02
PetType 0.01 1.0 0.03 -0.02 -0.0 -0.01 0.01 -0.05 -0.01 0.01 0.13 0.14 -0.77 -0.17 0.13 0.77 -0.17 -0.01 0.01 0.02 -0.04 0.02
PetID 1.0 0.01 -0.01 0.01 0.03 0.01 0.01 0.02 0.01 0.0 0.05 0.0 -0.03 -0.01 -0.02 0.03 0.01 0.02 -0.01 -0.02 0.03 -0.0 0.0
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6

```
data1 = data1.drop(['PetType'], axis = 1)
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2007 entries, 0 to 2006
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   PetID                   2007 non-null   int64
 1   AgeMonths               2007 non-null   int64
 2   Size                    2007 non-null   int64
 3   WeightKg                2007 non-null   float64
 4   Vaccinated              2007 non-null   int64
 5   HealthCondition         2007 non-null   int64
 6   TimeInShelterDays       2007 non-null   int64
 7   AdoptionFee             2007 non-null   int64
 8   PreviousOwner           2007 non-null   int64
 9   AdoptionLikelihood      2007 non-null   int64
 10  Breed_Golden Retriever  2007 non-null   int64
 11  Breed_Labrador          2007 non-null   int64
 12  Breed_Parakeet          2007 non-null   int64
 13  Breed_Persian           2007 non-null   int64
 14  Breed_Poodle            2007 non-null   int64
 15  Breed_Rabbit            2007 non-null   int64
 16  Breed_Siamese           2007 non-null   int64
 17  Color_Black             2007 non-null   int64
 18  Color_Brown             2007 non-null   int64
 19  Color_Gray              2007 non-null   int64
 20  Color_Orange            2007 non-null   int64
 21  Color_White             2007 non-null   int64
dtypes: float64(1), int64(21)
memory usage: 345.1 KB
```

Q2. Impact of L1, L2, and elastic net regularization on linear regression coefficients, performance, and interpretability.

Linear Regression:

Linear Regression is a type of regression methods used to understand a linear relationship between the inputs and output variables. It is one of the powerful techniques in machine learning that helps in predicting the values of variables based on inputs and outputs.This method evaluates the best fitting by minimizing the difference between the predicted values and actual values in the dataset.

In linear regression we try to find the optimal value of Beta (β) associated with each independent variable and the line, plane or hyperplane that will help us in getting optimal beta values will be considered as best fit.

Equation:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + ...\beta nxn$$

Here,

y= Dependent variable
β0,β1,βn = Intercepts
x1,x2,xn = Independent variables

In our code we have used Regularization to prevent overfitting by adding a penalty to the model's cost function. This will help in controlling the complexity of the model by discouraging overly large weights/coefficients.

Linear Regression Without Regularization:

In this step we have used two parameters in SGD Regressor: **random_state as 1 and Penalty as None.**

After this, we have used 'eta0' (learning rate) and 'max_iter' (maximum number of iterations) as hyperparameters and passed the suitable values to them.
The next step we have followed is by passing the above mentioned parameters in GridSearchCV. To score our regression model we have used the scoring as r^2 and CV=5 which implies 5-fold cross validation.

Here is the snippet of the code:

```
# Linear Regression (LR)

LinearRegression1 = SGDRegressor(random_state = 1, penalty = None) # building
Hparameter1 = {'eta0': [.0001, .001, .01, .1, 1,2,3,4], 'max_iter':[10000, 20000, 30000, 40000,50000,60000]}
grid_search1 = GridSearchCV(estimator=LinearRegression1, param_grid=Hparameter1, scoring='r2', cv=5)
grid_search1.fit(X_scaled, Y)

# results = DataFrame.from_dict(grid_search1.cv_results_)
# print("Cross-validation results:\n", results)
best_parameters = grid_search1.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search1.best_score_
print("Best result: ", best_result)
best_model = grid_search1.best_estimator_
print("Intercept β0: ", best_model.intercept_)
#print(DataFrame(zip(X.columns, best_model.coef_), columns=['Features','Coefficients']))
print(DataFrame(zip(X.columns, best_model.coef_), columns=['Features','Coefficients']).sort_values(by=['Coefficients'],ascending=False))
```

The Output after implementing the above code is as follows:

Best parameters: {'eta0': 0.01, 'max_iter': 10000} Best result: 0.23188474646791774

Intercept β0: [0.32492034]

Feature and Coefficients without regularization:

	Features	Coefficients
5	Vaccinated	0.139804
11	Breed_Labrador	0.112604
0	PetID	0.022526
7	TimeInShelterDays	0.016407
1	PetType	0.011783
20	Color_Orange	0.009527
14	Breed_Poodle	0.005588
18	Color_Brown	0.005364
3	Size	0.003774
19	Color_Gray	0.003161
4	WeightKg	-0.000437
17	Color_Black	-0.003903
8	AdoptionFee	-0.005591
16	Breed_Siamese	-0.009584
21	Color_White	-0.013906
9	PreviousOwner	-0.014426
10	Breed_Golden Retriever	-0.014495
13	Breed_Persian	-0.016210
12	Breed_Parakeet	-0.020790
15	Breed_Rabbit	-0.030961
2	AgeMonths	-0.093315
6	HealthCondition	-0.114642

To create a baseline model, I began using standard linear regression (no regularization). With the use of this baseline, models using regularization techniques like L1 (Lasso), L2 (Ridge), and Elastic Net can be clearly compared.By first understanding the improvements in performance, generalization, and interpretability that regularization brought about more fully if I first understood the unregularized model's performance and coefficient estimates. Starting with a full-feature model also aids in emphasizing the importance of every feature before regularization penalizes or eliminates any of them. A complete grasp of the model development process and the advantages of regularization is ensured by this methodical approach.

Without regularization, the model attempts to fit the data as closely as possible. This often results in larger coefficients, which can lead to overfitting, especially if the dataset has high-dimensional features.

Note: when we use regularization techniques, we will be considering the hyperparameters values we got from the previous step and then we are tuning our model accordingly.

Linear Regression With Regularization:

To evaluate the impact on our model, we have used all the 3 regularizations techniques: L1, L2 and elastic net.

1. L1(Lasso): Lasso uses a penalty term to minimize the coefficients of less important independent variables to zero on training data to reduce model complexity.
In this technique we have used 3 hyperparameters : ‘eta0’, ‘max_iter’ and ‘alpha’. The value of eta0 and max_iter we have tuned with the help of the previous step(Linear Regression without regularization).

```
# #  Regularization - L1
LinearRegression1 = linear_model.SGDRegressor(random_state = 1, penalty = 'l1') # model building
Hparameter2 = {'eta0': [.0001, .001, .01, .1, 1], 'max_iter':[10000, 20000, 30000, 40000],'alpha': [.001, .01, .1, 1,10, 100]}

grid_search2 = GridSearchCV(estimator=LinearRegression1, param_grid=Hparameter2, scoring='r2', cv=5)
grid_search2.fit(X_scaled, Y)

# results = DataFrame.from_dict(grid_search2.cv_results_)
# print("Cross-validation results:\n", results)
best_parameters = grid_search2.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search2.best_score_
print("Best result: ", best_result)
best_model = grid_search2.best_estimator_
print("Intercept β0: ", best_model.intercept_)
print(DataFrame(zip(X.columns, best_model.coef_), columns=['Features','Coefficients']).sort_values(by=['Coefficients'],ascending=False))
```

Best parameters: {'alpha': 0.01, 'eta0': 0.01, 'max_iter': 10000} Best result: 0.24101368920904181 Intercept β0: [0.32519329]

Feature and Coefficients without regularization:

```
Best parameters:  {'alpha': 0.01, 'eta0': 0.01, 'max_iter': 10000}
Best result:  0.24101368920904181
Intercept β0:  [0.32519329]
                  Features  Coefficients
5               Vaccinated      0.128300
11          Breed_Labrador      0.114800
0                    PetID      0.012527
3                     Size      0.000000
4                 WeightKg      0.000000
20            Color_Orange      0.000000
19              Color_Gray      0.000000
18             Color_Brown      0.000000
17             Color_Black      0.000000
16           Breed_Siamese      0.000000
14            Breed_Poodle      0.000000
13           Breed_Persian      0.000000
12          Breed_Parakeet      0.000000
1                   PetType      0.000000
10  Breed_Golden Retriever      0.000000
9            PreviousOwner      0.000000
8              AdoptionFee      0.000000
7        TimeInShelterDays      0.000000
21            Color_White     -0.003776
15            Breed_Rabbit     -0.004622
2               AgeMonths     -0.082642
6          HealthCondition     -0.104576
```

2. L2(Ridge): Ridge used to prevent overfitting in linear regression with multiple independent variables.The hyperparameters we have used here are the same as L1 but we have tuned them more after training our model on L1. Below are the results:

```
# #  Regularization - L2
LinearRegression2 = linear_model.SGDRegressor(random_state = 1, penalty = 'l2') # model building
Hparameter2 = {'eta0': [.0001, .001, .01, .1, 1], 'max_iter':[10000, 20000, 30000, 40000],'alpha': [.001, .01, .1, 1,10, 100]}

grid_search2 = GridSearchCV(estimator=LinearRegression2, param_grid=Hparameter2, scoring='r2', cv=5)
grid_search2.fit(X_scaled, Y)

# results = DataFrame.from_dict(grid_search2.cv_results_)
# print("Cross-validation results:\n", results)
best_parameters = grid_search2.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search2.best_score_
print("Best result: ", best_result)
best_model = grid_search2.best_estimator_
print("Intercept β0: ", best_model.intercept_)
print(DataFrame(zip(X.columns, best_model.coef_), columns=['Features','Coefficients']).sort_values(by=['Coefficients'],ascending=False))
```

Best parameters: {'alpha': 0.1, 'eta0': 0.01, 'max_iter': 10000} Best result: 0.23288752152643588 Intercept β0: [0.32489075]

Feature and Coefficients L2:

	Features	Coefficients
5	Vaccinated	0.126882
11	Breed_Labrador	0.104224
0	PetID	0.019644
7	TimeInShelterDays	0.015831
1	PetType	0.010981
20	Color_Orange	0.008698
18	Color_Brown	0.006069
14	Breed_Poodle	0.005612
3	Size	0.004699
19	Color_Gray	0.002542
4	WeightKg	0.000452
17	Color_Black	-0.003186
8	AdoptionFee	-0.004177
16	Breed_Siamese	-0.008849
21	Color_White	-0.013857
10	Breed_Golden Retriever	-0.013943
9	PreviousOwner	-0.014571
13	Breed_Persian	-0.014983
12	Breed_Parakeet	-0.019312
15	Breed_Rabbit	-0.028569
2	AgeMonths	-0.085166
6	HealthCondition	-0.104300

Elastic net: This is a combination of both L1 and L2 regularization that combines penalties of both Lasso and Ridge regression into a single model. There are 4 hyperparameters we have used here: ‘eta0’, ‘max_iter’, ‘alpha’ and ‘L1 Ratio’

Best parameters: {'alpha': 0.01, 'eta0': 0.01, 'l1_ratio': 0.9, 'max_iter': 10000} Best result: 0.24101411223857658 Intercept β0: [0.32518114]

Feature and Coefficients L2:

	Features	Coefficients
5	Vaccinated	0.129159
11	Breed_Labrador	0.115560
0	PetID	0.013511
3	Size	0.000000
4	WeightKg	0.000000
20	Color_Orange	0.000000
19	Color_Gray	0.000000
18	Color_Brown	0.000000
17	Color_Black	0.000000
16	Breed_Siamese	0.000000
14	Breed_Poodle	0.000000
13	Breed_Persian	0.000000
12	Breed_Parakeet	0.000000
1	PetType	0.000000
10	Breed_Golden Retriever	0.000000
9	PreviousOwner	0.000000
8	AdoptionFee	0.000000
7	TimeInShelterDays	0.000000
15	Breed_Rabbit	-0.005379
21	Color_White	-0.005738
2	AgeMonths	-0.083433
6	HealthCondition	-0.105424

Interpretation :

Modified_r2 = 1 - (1 - best_result) * (1 - (4/5*r) / (4/5*r - c - 1)

When dealing with high-dimensional data, this computation modifies the R2 score to take sample size and feature count into consideration, yielding a more precise performance indicator. The updated R2 for your final Elastic Net regularized model was 0.2305, which helps to make sure that, considering the amount of predictors, the model's performance evaluation is not too optimistic.

Regularization techniques had a considerable impact on interpretability, performance, and coefficient shrinkage in our linear regression model. The baseline R2 score for standard linear regression without regularization was 23% at first. Upon applying L1 regularization (Lasso), some coefficients were lowered to zero, such as 'Size,' 'WeightKg,' and different 'Color' and 'Breed' features. This resulted in an increase in the model's interpretability while keeping the R^2 score at 24%. Similar R^2 score of 23% was obtained using L2 regularization (Ridge), which shrank the coefficients without removing any features. With an R2 score of 24%, Elastic Net regularization—which combines the L1 and L2 effects—performed the best while maintaining interpretability by decreasing some coefficients to zero. Overall, the model's performance and interpretability were somewhat enhanced by regularization, particularly Elastic Net, while maintaining a negligible loss in predictive capability.Hence, after analysis we can say that our model was not overfitted at the first step itself. But with the help of the elastic net we can see that our performance has increased slightly.

Q3. Impact of L2 regularization on support vector regression performance and interpretability.

Support Vector Regression as the name suggests is a regression algorithm that supports both linear and non-linear regressions.

The L2 regularization also known as the Ridge has a major role in terms of both performance and Interpretability. It basically reduces the weights towards zero which eventually help in preventing overfitting.

In SVR, we have used 2 hyperparameters : Regularization Parameter ‘C’ and ‘Kernel’. Their values we have chosen in code as:

```
#Support Vector Regression
from sklearn.svm import SVR
SVRegressor = SVR()
Hparameters = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [10,30,60,90,110]}
grid_search1 = GridSearchCV(estimator=SVRegressor, param_grid=Hparameters, scoring='r2', cv=5)
grid_search1.fit(X_, Y)

best_parameters = grid_search1.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search1.best_score_
print("Best result: ", best_result)
```

```
Best parameters:  {'C': 10, 'kernel': 'rbf'}
Best result:  0.5273304731074328
```

Kernel: ['linear', 'poly', 'rbf', 'sigmoid'] 'C': [10,30,60,90,110]

After, implementing the SVR with the above mentioned hyperparameters we have received the following output:
Best Parameters: {‘C’:10, ‘Kernel’: rbf}
r2: 0.5273304731074328

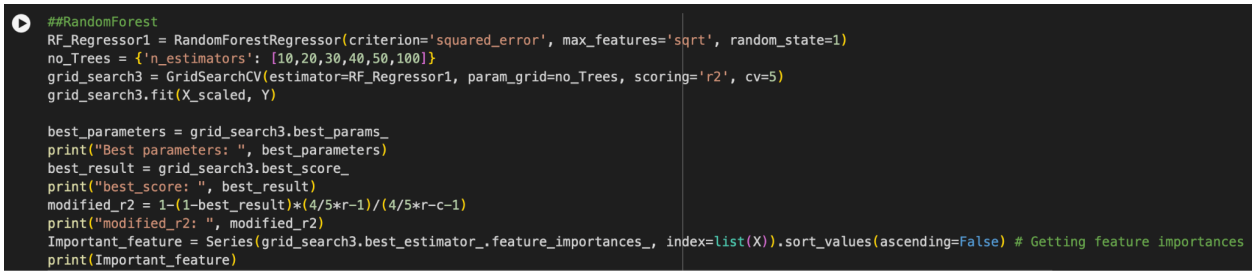
With a rbf kernel and a high value of C (10), the Support Vector Regression (SVR) model performed at its peak, producing an R2 score of 0.527. The model may be more prone to overfitting because of this high C value, which indicates that the model is well-suited to the training set. The SVR model's performance is not as striking as that of the Random Forest and Linear Regression models, which showed higher prediction accuracy. In addition, the SVR model adds additional complexity and decreases interpretability, which makes it a less desirable option when considering the more straightforward and efficient models in our research.

4. If you were to implement random forest regression, then its comparative performance and interpretability with respect to regularized linear regression and regularized support vector regression models.

Random forest regression involves splitting the decision tree's nodes in such a way as to reduce the error in the resulting nodes, rather than reducing the entropy, since no classes are involved in regression, both the left and right nodes of the decision tree will have their own errors.

In SVR, the use of a support vector is used to define a linear regression. Continuous linear regression enhances traditional linear regression by incorporating regularization methods such as L1 (Lasso) and L2 (Ridge).

The hyperparameters that will be tuned are n_estimators and the metric for the evaluation is r2.



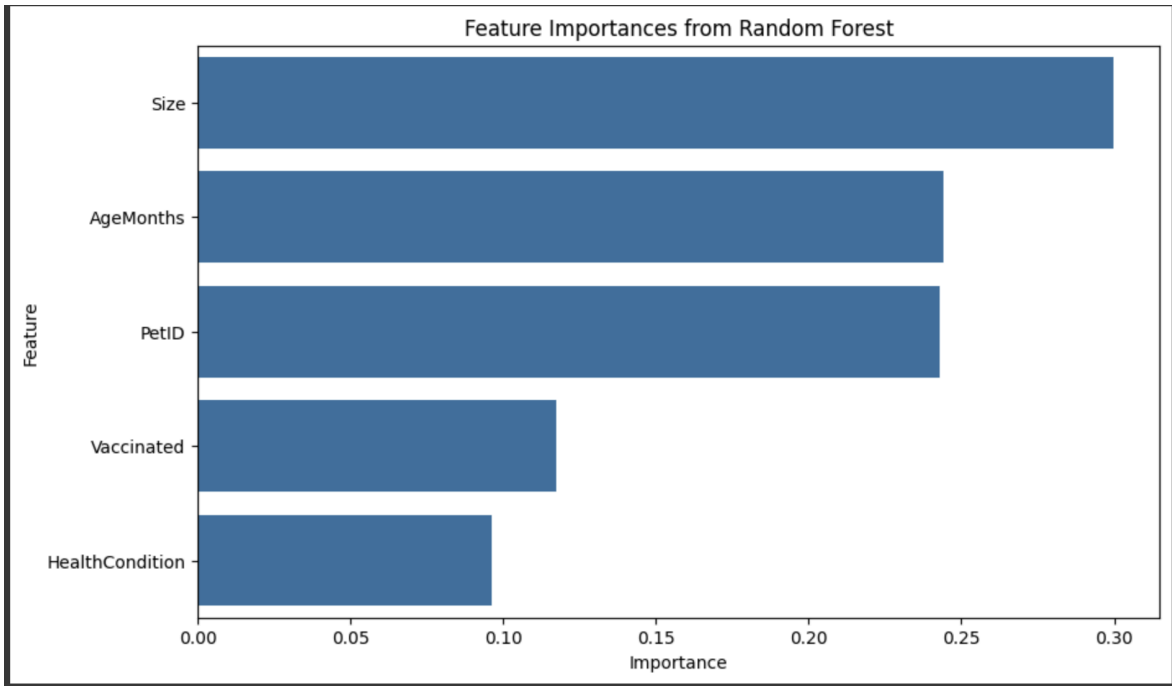
```
##RandomForest
RF_Regressor1 = RandomForestRegressor(criterion='squared_error', max_features='sqrt', random_state=1)
no_Trees = {'n_estimators': [10,20,30,40,50,100]}
grid_search3 = GridSearchCV(estimator=RF_Regressor1, param_grid=no_Trees, scoring='r2', cv=5)
grid_search3.fit(X_scaled, Y)

best_parameters = grid_search3.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search3.best_score_
print("best_score: ", best_result)
modified_r2 = 1-(1-best_result)*(4/5*r-1)/(4/5*r-c-1)
print("modified_r2: ", modified_r2)
Important_feature = Series(grid_search3.best_estimator_.feature_importances_, index=list(X)).sort_values(ascending=False) # Getting feature importances
print(Important_feature)
```

Output values for random forest : When we implement the random forest method , we get the following results.

Best parameters: {'n_estimators': 100} best_score: best_score: 0.6495780640362635 modified_r2: 0.6447067872820602

```
Size                      0.212913
AgeMonths                 0.151379
PetID                     0.109824
Vaccinated                0.102113
HealthCondition           0.070878
AdoptionFee               0.066904
WeightKg                  0.065106
Breed_Labrador            0.064077
TimeInShelterDays         0.059798
PetType                   0.016417
PreviousOwner             0.010973
Color_White               0.008342
Color_Black               0.008051
Color_Gray                0.007773
Color_Orange              0.007565
Color_Brown               0.007113
Breed_Rabbit              0.005903
Breed_Golden Retriever    0.005628
Breed_Parakeet            0.005163
Breed_Poodle              0.004878
Breed_Siamese             0.004686
Breed_Persian             0.004516
dtype: float64
```

```
# Selecting features with higher sifnificance and redefining feature set
X = data1[['Size', 'AgeMonths', 'PetID', 'Vaccinated','HealthCondition']]
X_scaled = StandardScaler().fit_transform(X)

RF_Regressor2 = RandomForestRegressor(criterion='squared_error', max_features='sqrt', random_state=1)
no_Trees = {'n_estimators': [100,200,300,400,500]}
grid_search4 = GridSearchCV(estimator=RF_Regressor2, param_grid=no_Trees, scoring='r2', cv=10)
grid_search4.fit(X_scaled, Y)

best_parameters = grid_search4.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search4.best_score_
print("r2: ", best_result)
modified_r2 = 1-(1-best_result)*(4/5*r-1)/(4/5*r-c-1)
print("modified_r2: ", modified_r2)
best_model = grid_search4.best_estimator_
```



After implementing feature selection for the random forest regression ,when the first 5 features with higher significance are taken into consideration then the score decreases to r2: 0.5965405922912747 from 0.6495780640362635.

Output :

Best parameters: {n_estimators': 500} r2: 0.596540592291274
modified_r2: 0.5909320323458734

The Random Forest Regressor significantly outperforms the Support Vector Regressor (SVR) in this analysis. An R2 score of 0.649 and a modified R2 score of 0.6447 are obtained by the Random Forest model, which was refined using 500 estimators. This suggests a good match to

the data. After implementing feature selection for the random forest regression ,when the first 5 features with higher significance are taken into consideration then the score decreases in r2. A small difference in performance, especially when simplifying the model by using fewer features, can be acceptable. Sometimes, a slightly less accurate model that is simpler and more interpretable might be preferred over a more complex model with slightly better performance.

It's important to consider the trade-off between model accuracy and complexity. If the reduction in performance is small and the simpler model offers benefits such as improved interpretability or reduced computational cost, it might be a worthwhile compromise.

On the other hand, the SVR model with the optimal parameters (kernel='rbf' and C=10) produces a lower R2 score of 0.527. This demonstrates that the SVR model may struggle with the complexity of the data, perhaps overfitting with a high C value, and that its performance is less effective than that of the Random Forest model.

Therefore, as compared to the SVR model, the Random Forest Regressor not only delivers a better fit and higher performance, but it also offers more interpretability and is less prone to overfitting. It is the better option in this analysis since it captures non-linear correlations between AdoptionLikelihood and the independent variables.

Q5. Performing a prediction for one of your models using new data

We arrive at the following conclusions after assessing and optimizing a number of regression models on the adoption dataset, including Linear Regression, Regularized Linear Regression (L1, L2, and Elastic Net), Support Vector Regression (SVR), and Random Forest Regression:

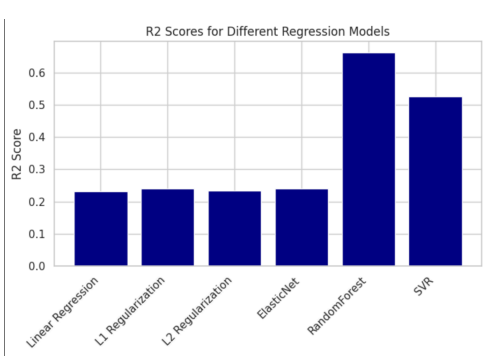
Linear Regression and Regularized Models: With R^2 scores ranging from 0.231 to 0.241, the linear models, namely those with L1 (Lasso), L2 (Ridge), and Elastic Net regularization, showed rather steady performance. These models showed only slight regularization improvements and did not significantly outperform one another. Although regularization contributed to the selection of features and eliminated some coefficients, the overall performance of the model stayed very low.

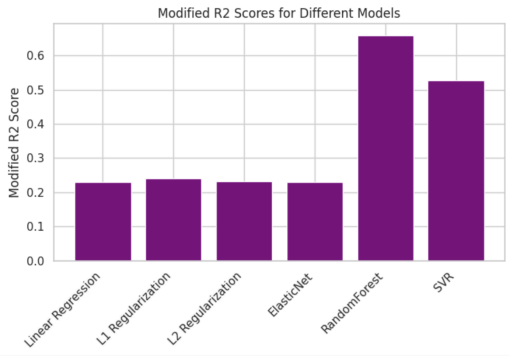
Support Vector Regression (SVR): An optimal R2 score of 0.527 was obtained by the SVR model. Even with hyperparameter optimization, SVR's performance fell short of other models'. The complexity of the model makes it more difficult to interpret, and the high value of parameter C points to a risk of overfitting.

Random Forest Regression: The Random Forest Regressor's original R^2 score of 0.649, and its corresponding adjusted R^2 score 0.644, making it the most successful model. This model successfully captured a sizable percentage of the variance in the objective variable, AdoptionLikelihood, by employing the top five features. The model's greater fit and

interpretability are shown by the improvement in performance and its capacity to manage non-linear relationships.

In conclusion, compared to linear regression and SVR, the Random Forest Regressor offers a higher R2 score and improved interpretability, making it the best-fitting model for this dataset. The Random Forest model is more effective than other regression models in this research because it can capture a significant amount of variance with fewer features.






```
Random Forest Best Parameters: {'n_estimators': 500}
Random Forest R2 Score: 0.6495
Random Forest Modified R2 Score: 0.6447

Feature Importance from Random Forest:
           Feature  Importance
0             Size    0.300030
2            PetID    0.243633
1        AgeMonths    0.242524
3       Vaccinated    0.117615
4  HealthCondition    0.096198

SVR Best Parameters: {'C': 10, 'kernel': 'rbf'}
SVR R2 Score: 0.5273
```

Conclusion for 🐾 Paws & Claws Adoption Center 🐾 as Business Analysts -

Focus on Key Features: The adoption center should prioritize the top features identified by the Random Forest model when developing strategies to improve adoption rates. Treating health concerns and breed-specific issues, for instance, may increase the likelihood of adoption.

Put Data-Driven Strategies into Practice: Utilize the Random Forest model's data to create focused interventions, such as better care guidelines for pets with lower adoption chances or specialized marketing campaigns for pets with higher adoption potential.

Ongoing Observation: To keep the model current and correct, add new data to it on a regular basis. This will support the adoption center's ongoing efforts to enhance adoption tactics and adjust to shifting trends.

INDIVIDUAL CONTRIBUTIONS TO THE ASSIGNMENT :

DATASET SELECTION - Simran shruthi ,Sunil Rao
CODE IMPLEMENTATION -

Data preparation, SVM,Visualization - SimranShruthi

LR and RFR- Sunil Rao

ANALYSIS OF REGRESSION MODELS AND CONCLUSION -

Simran shruthi ,Sunil Rao

** THE END**