About me:

YAHYA HAFFAR

ReDI School of Digital Integration

LinkedIn:

Predicting Car Insurance Claims

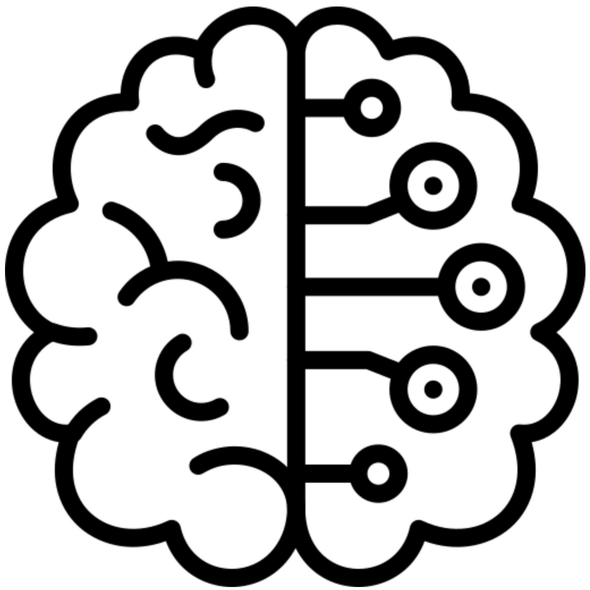
Problem:
The insurance company would like to optimize its insurance policy.
Therefore It would like to estimate the likelihood that customers will make a claim.

Proposed Solution:
Create and deploy a machine learning Model that can predict whether a customer will make a claim on their insurance during the policy period.

Predicting whether a client will make insurance claims

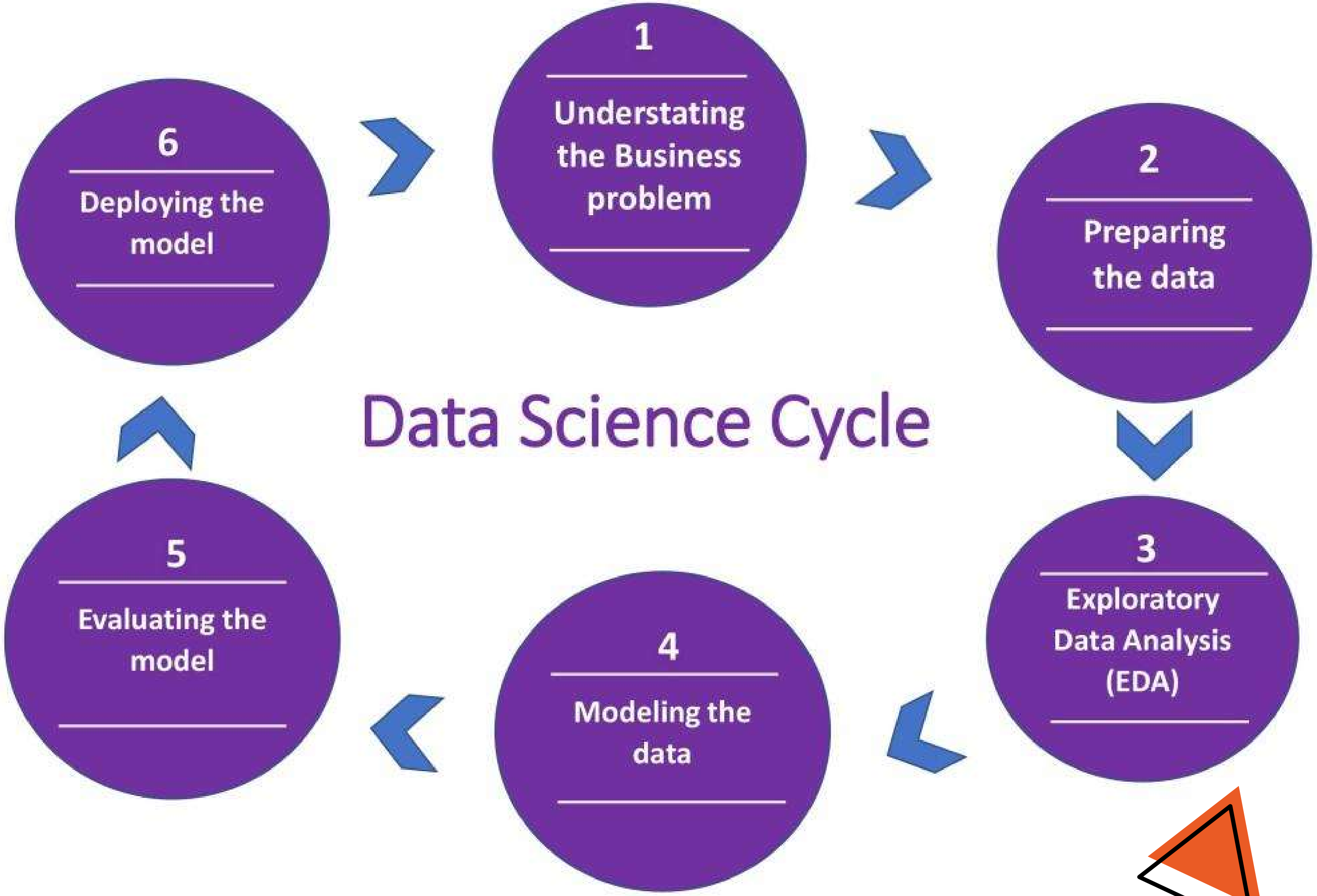
Features

- age
- driving_experience
- vehicle_ownership
- Children
- ….



Model

Output:
Will make a Claim
Won’t make a claim


Data Science Cycle
1
Understating the Business problem
2
Preparing the data
3
Exploratory Data Analysis (EDA)
4
Modeling the data
5
Evaluating the model
6
Deploying the model

Data Preparation:

	id	age	gender	driving_experience	education	income	credit_score	vehicle_ownership	vehicle_year	married	children	postal_code	annual_mileage
0	569520	3	0	0-9y	high school	upper class	0.629027	1.0	after 2015	0.0	1.0	10238	12000.0
1	750365	0	1	0-9y	none	poverty	0.357757	0.0	before 2015	0.0	0.0	10238	16000.0
2	199901	0	0	0-9y	high school	working class	0.493146	1.0	before 2015	0.0	0.0	10238	11000.0
3	478866	0	1	0-9y	university	working class	0.206013	1.0	before 2015	0.0	1.0	32765	11000.0
4	731664	1	1	10-19y	none	working class	0.388366	1.0	before 2015	0.0	0.0	32765	12000.0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   10000 non-null  int64
 1   age                  10000 non-null  int64
 2   gender               10000 non-null  int64
 3   driving_experience   10000 non-null  object
 4   education            10000 non-null  object
 5   income               10000 non-null  object
 6   credit_score         9018 non-null   float64
 7   vehicle_ownership    10000 non-null  float64
 8   vehicle_year         10000 non-null  object
 9   married              10000 non-null  float64
 10  children             10000 non-null  float64
 11  postal_code          10000 non-null  int64
 12  annual_mileage       9043 non-null   float64
 13  vehicle_type         10000 non-null  object
 14  speeding_violations  10000 non-null  int64
 15  duis                 10000 non-null  int64
 16  past_accidents       10000 non-null  int64
 17  outcome              10000 non-null  float64
dtypes: float64(6), int64(7), object(5)
```

```
categorical_cols = ['driving_experience', 'education', 'income', 'vehicle_year' , 'vehicle_type']
cars[categorical_cols] = cars[categorical_cols].astype('category')
```

```
cars['credit_score'].fillna(cars['credit_score'].mean(), inplace = True)
cars['annual_mileage'].fillna(cars['annual_mileage'].mean(), inplace = True)
```

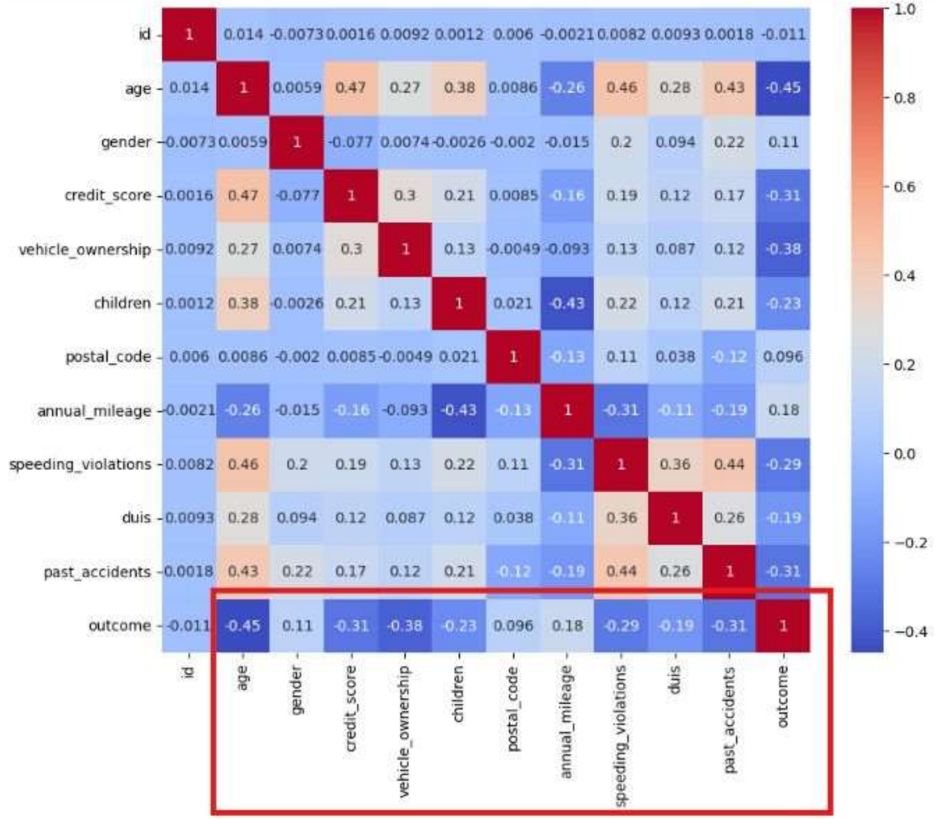
```
cars.info()
```

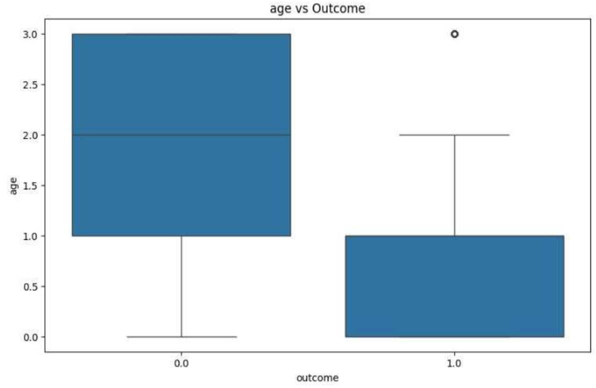
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   10000 non-null  int64
 1   age                  10000 non-null  int64
 2   gender               10000 non-null  int64
 3   driving_experience   10000 non-null  category
 4   education            10000 non-null  category
 5   income               10000 non-null  category
 6   credit_score         10000 non-null  float64
 7   vehicle_ownership    10000 non-null  float64
 8   vehicle_year         10000 non-null  category
 9   married              10000 non-null  float64
 10  children             10000 non-null  float64
 11  postal_code          10000 non-null  int64
 12  annual_mileage       10000 non-null  float64
 13  vehicle_type         10000 non-null  category
 14  speeding_violations  10000 non-null  int64
 15  duis                 10000 non-null  int64
 16  past_accidents       10000 non-null  int64
 17  outcome              10000 non-null  float64
dtypes: category(5), float64(6), int64(7)
memory usage: 1.0 MB
```

Data Exploration:

```python
# Calculate correlation matrix
corr = cars.corr()

# Plot correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```

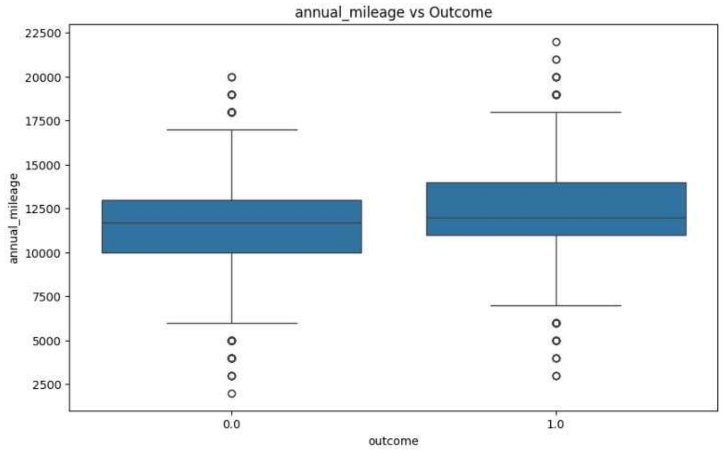
```
C:\Users\Senfinity\AppData\Local\Temp\ipykernel_33312\2199352778.py:2: FutureWarning: The default value of numeric_only in Data
Frame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
  corr = df.corr()
```





High correlation

Low correlation



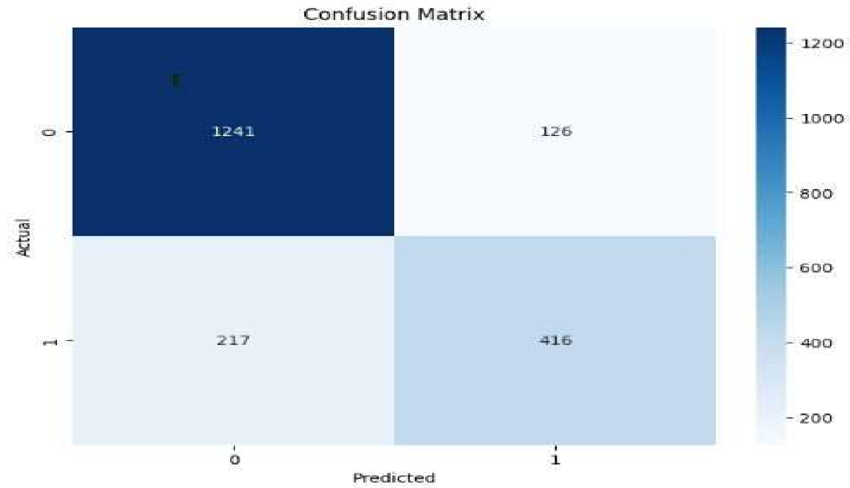
Data Modeling and Model Evaluation:

```python
# Separate the features (X) and target variable (y)
X = cars.drop('outcome', axis=1)
y = cars['outcome']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

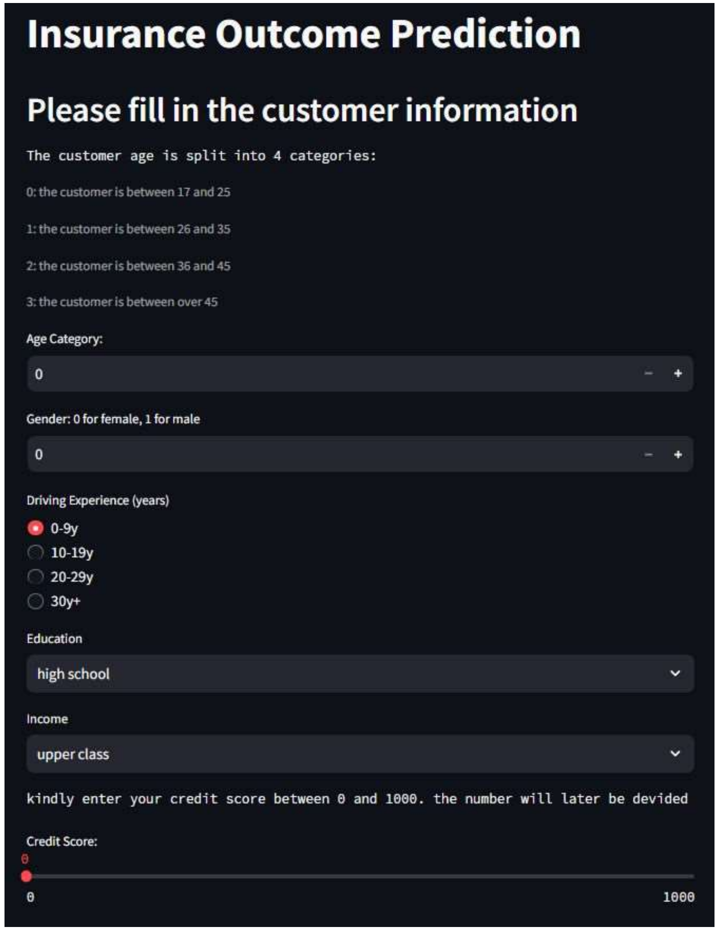
```python
# Create a list of categorical and numerical columns
categorical_cols = X_train.select_dtypes(include=['category']).columns
numerical_cols = X_train.select_dtypes(include=['int64', 'float64']).columns

# Create a ColumnTransformer to handle categorical and numerical columns separately
ct = ColumnTransformer([
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols),
    ('num', 'passthrough', numerical_cols)
])

# Create a pipeline with the ColumnTransformer and RandomForestClassifier
pipeline = Pipeline([
    ('transformer', ct),
    ('classifier', RandomForestClassifier())
])
```

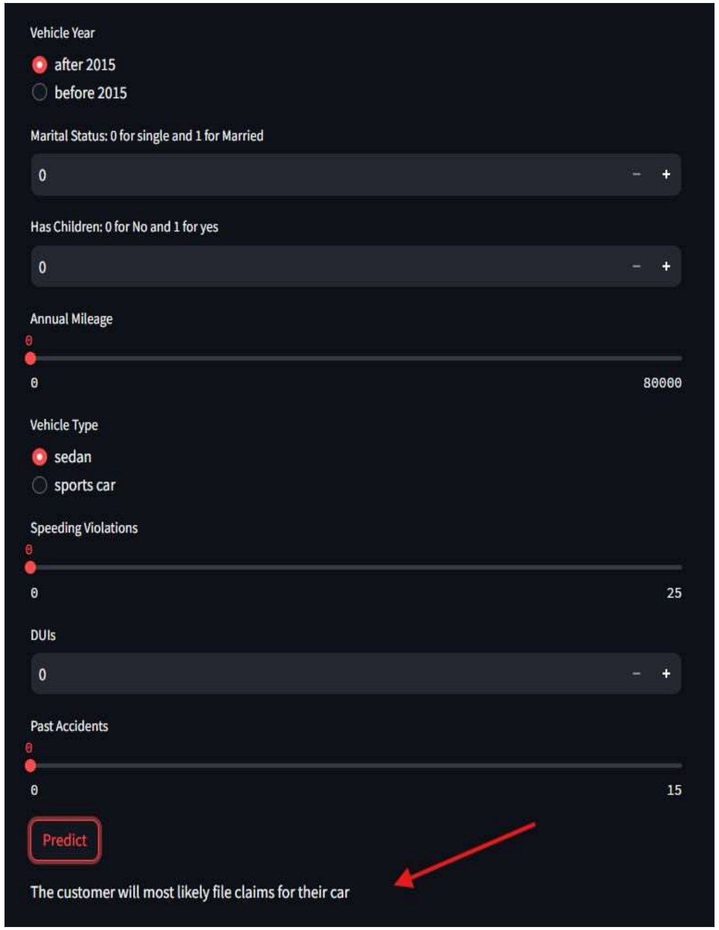
```python
# Train the pipeline
pipeline.fit(X_train, y_train)

# Make predictions on the test set
y_pred = pipeline.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(cm)


# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.8285
Confusion Matrix:
[[1241  126]
 [ 217  416]]
```



Model Deployment





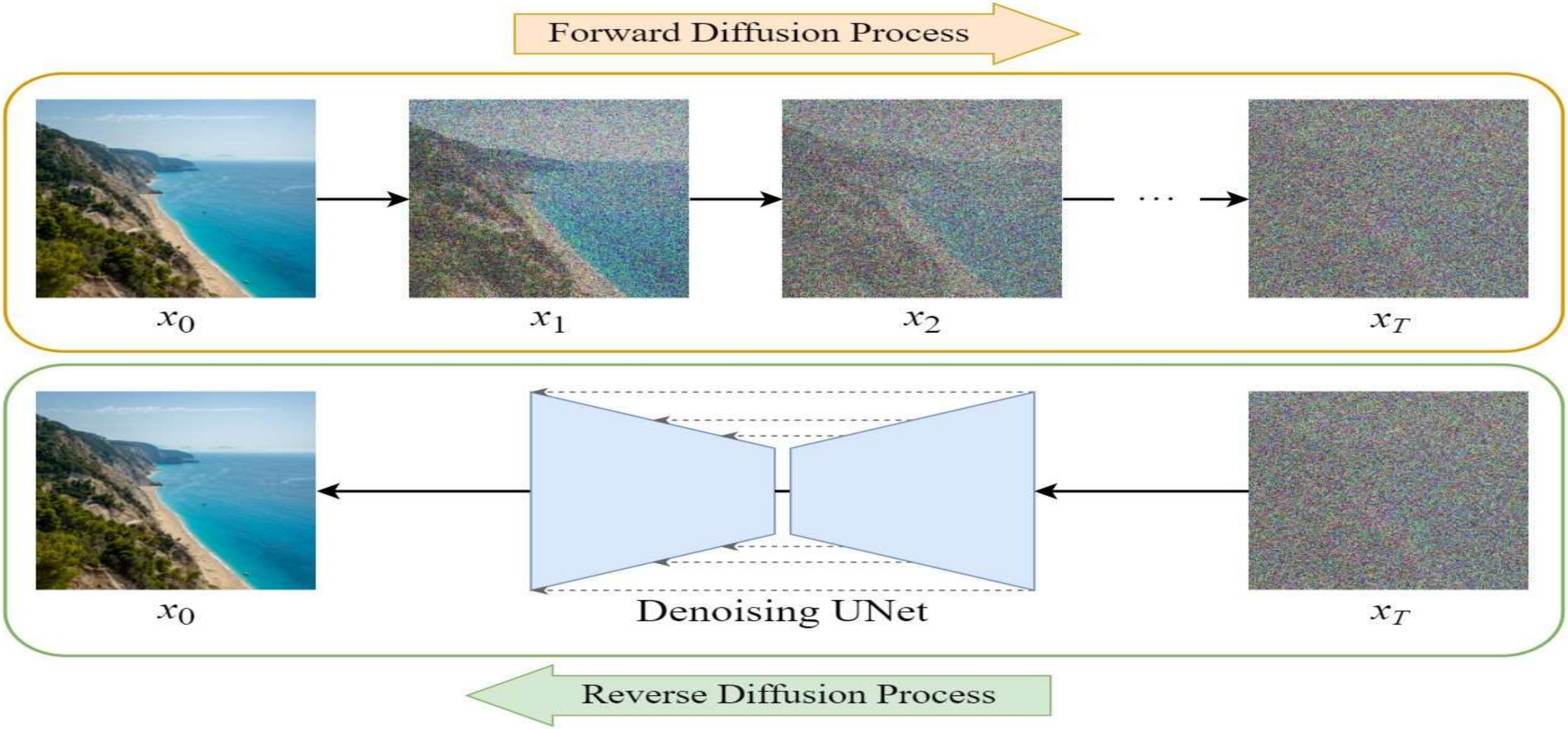
Using AI to Alter Images:

Image Inpainting:
Creating a mask for an image which is later used to add details to the image.

Diffusion Models: A diffusion model creates an image by gradually removing noise from random pixels, guided by a text description, until a clear picture emerges

Project: Create a web application that easily uses stable diffusion with inpainting, to alter images and add new components into them.

Simple Diffusion explanation:



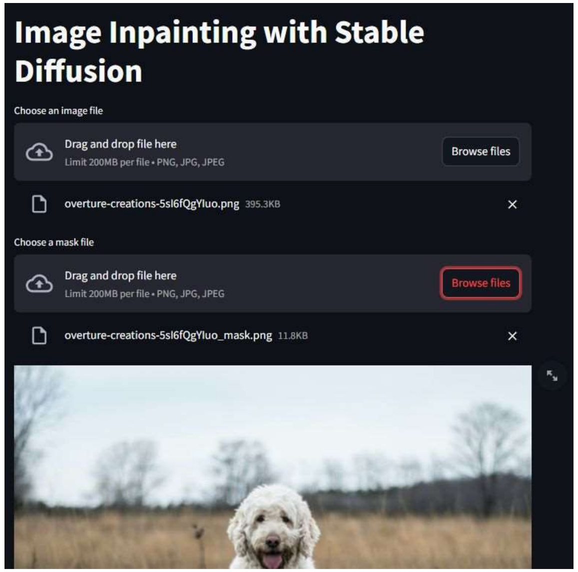
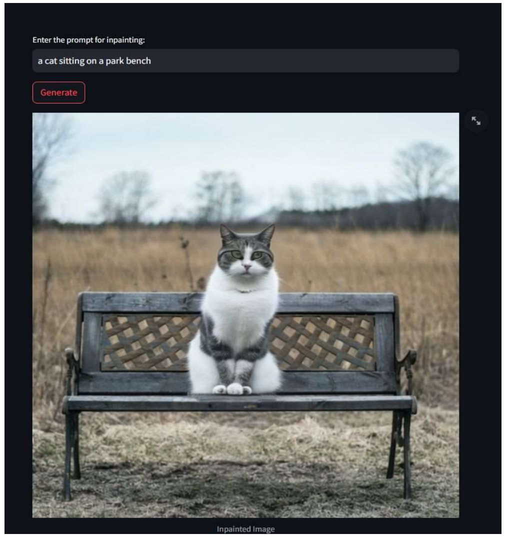
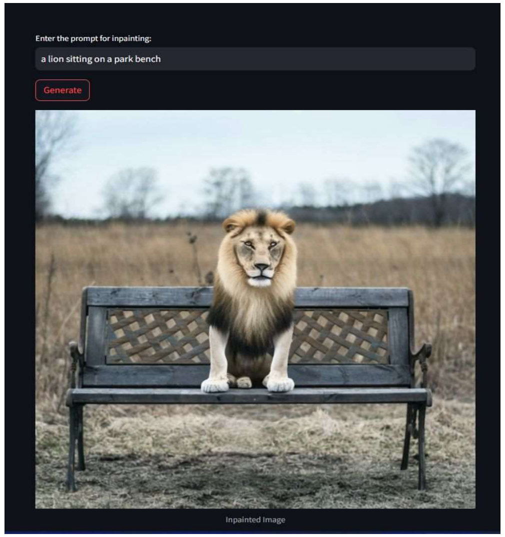
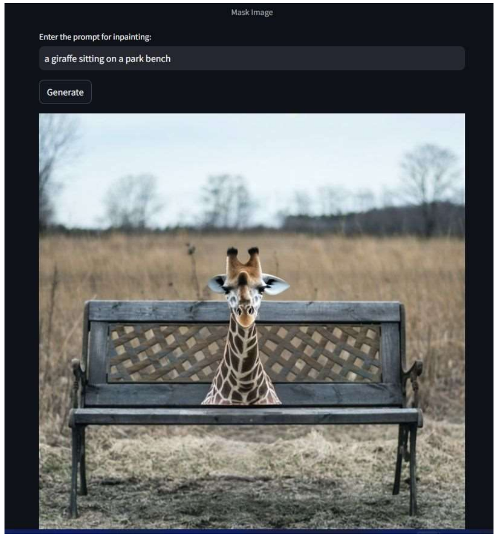


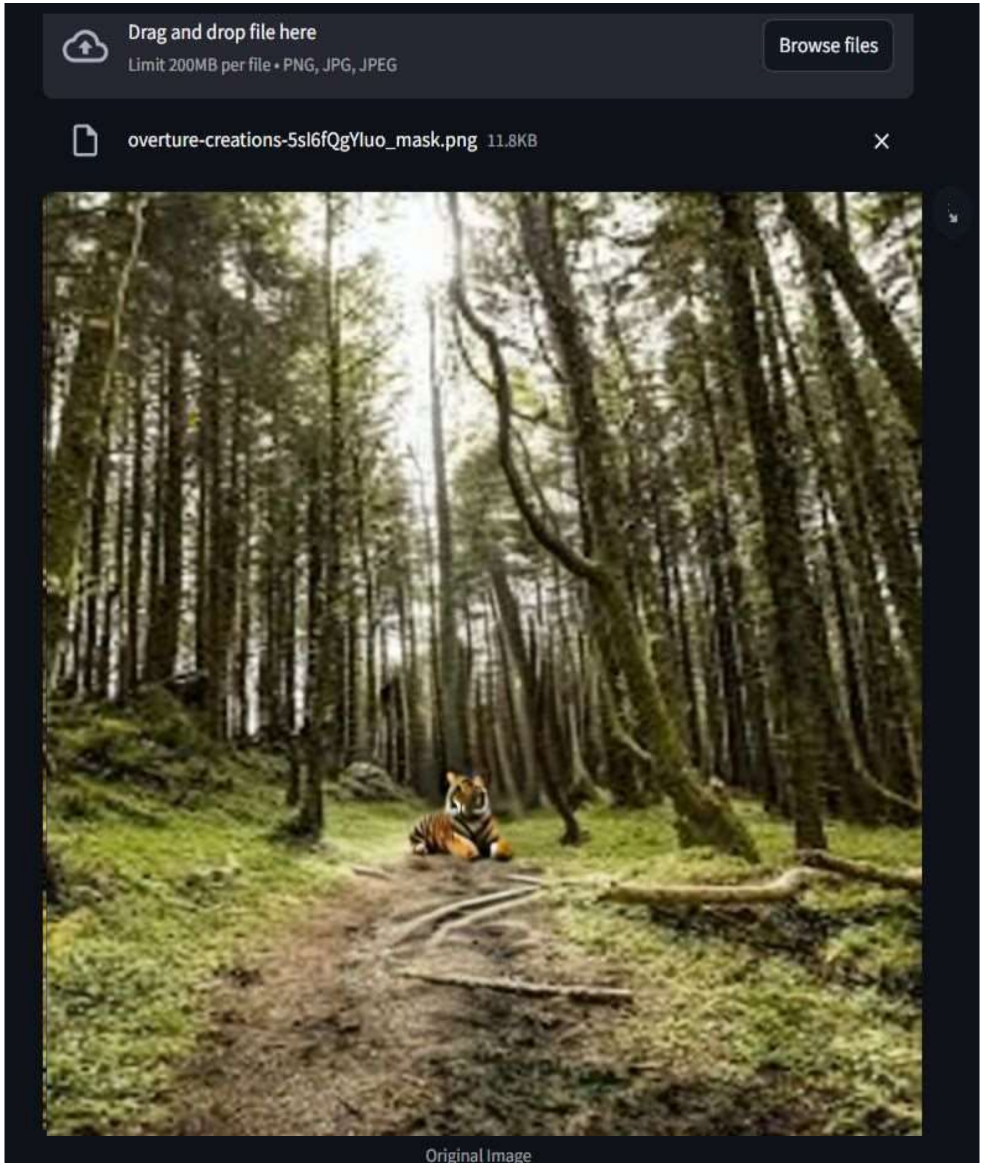
Image Inpainting with Stable Diffusion
Choose an image file
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files
overture-creations-5sI6fQgYIuo.png 395.3KB
Choose a mask file
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files
overture-creations-5sI6fQgYIuo_mask.png 11.8KB


Enter the prompt for inpainting:
a cat sitting on a park bench
Generate
Inpainted Image


Enter the prompt for inpainting:
a lion sitting on a park bench
Generate
Inpainted Image


Enter the prompt for inpainting:
an elephant sitting on a park bench
Generate
Inpainted Image


Mask Image
Enter the prompt for inpainting:
a giraffe sitting on a park bench
Generate


Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files
overture-creations-5sI6fQgYIuo_mask.png 11.8KB
Original Image

Tech Stack

ReDI SCHOOL

pandas

Streamlit

TensorFlow

colab

matplotlib

GitHub

git

torch

python™

Thank you

Let's Connect on LinkedIn