FaceHack: Predicting BMI & Gender

Thribhuvan S, EC21B1093

Ambati Lokesh, EC21B1104

Amar Rohith, EC21B1106

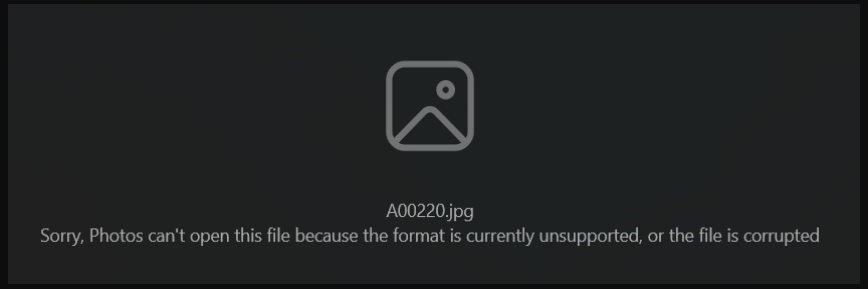
Introduction

- Objective: Using frontal and side-view facial images to predict BMI and Gender.
- Dataset: Mugshots of prison inmates, along with their personal information such as id, date of birth, height, weight, gender, race, hair & eye color.

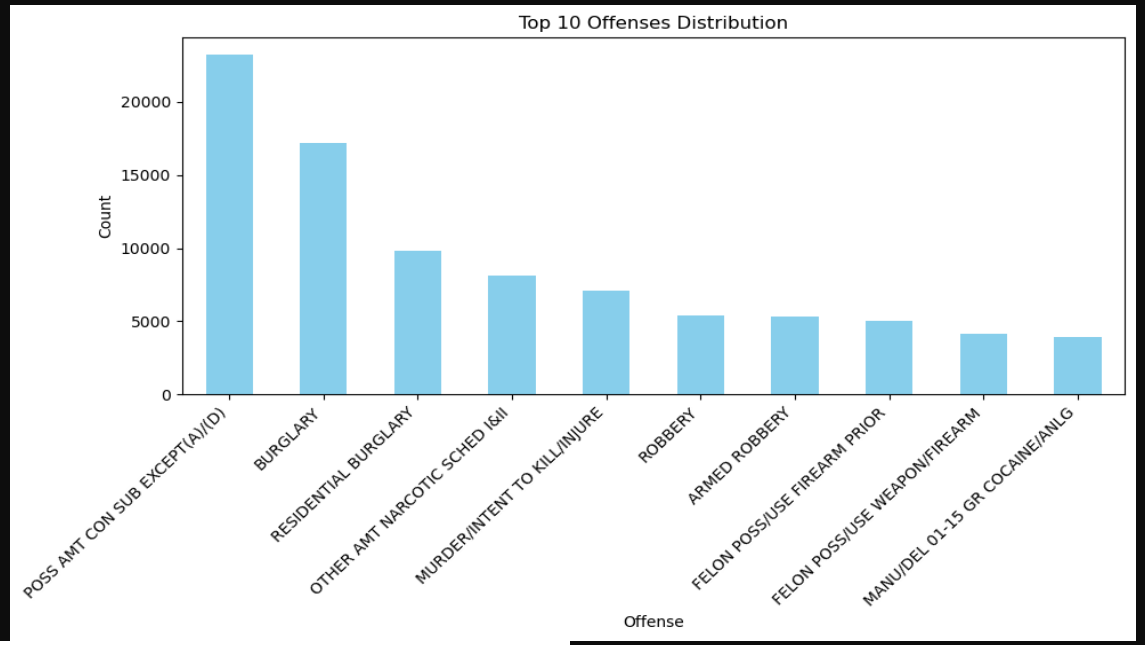
	id	date_of_birth	weight	hair	sex	height	race	eyes	admission_date
0	A00147	06/14/1949	185.0	Brown	Male	67.0	White	Blue	02/16/1983
1	A00220	03/30/1957	155.0	Black	Male	73.0	Black	Brown	05/19/2016
2	A00360	12/18/1946	167.0	Gray or Partially Gray	Male	69.0	White	Green	02/26/1988
3	A00367	01/12/1954	245.0	Black	Male	72.0	Black	Brown	11/09/2017
4	A01054	03/25/1954	166.0	Salt and Pepper	Male	67.0	Black	Brown	12/23/1988
...	...	...	...	...	...	...	...	...	...
61104	Y25362	05/30/1985	120.0	Blonde or Strawberry	Male	59.0	White	Brown	10/24/2017
61105	Y25363	05/15/1986	170.0	Brown	Male	71.0	White	Brown	10/25/2017
61106	Y25364	02/23/1972	112.0	Brown	Female	62.0	White	Green	10/25/2017
61107	Y25365	06/16/1992	158.0	Brown	Female	63.0	White	Brown	10/25/2017
61108	Y25366	03/04/1949	220.0	Gray or Partially Gray	Male	67.0	White	Blue	10/25/2017

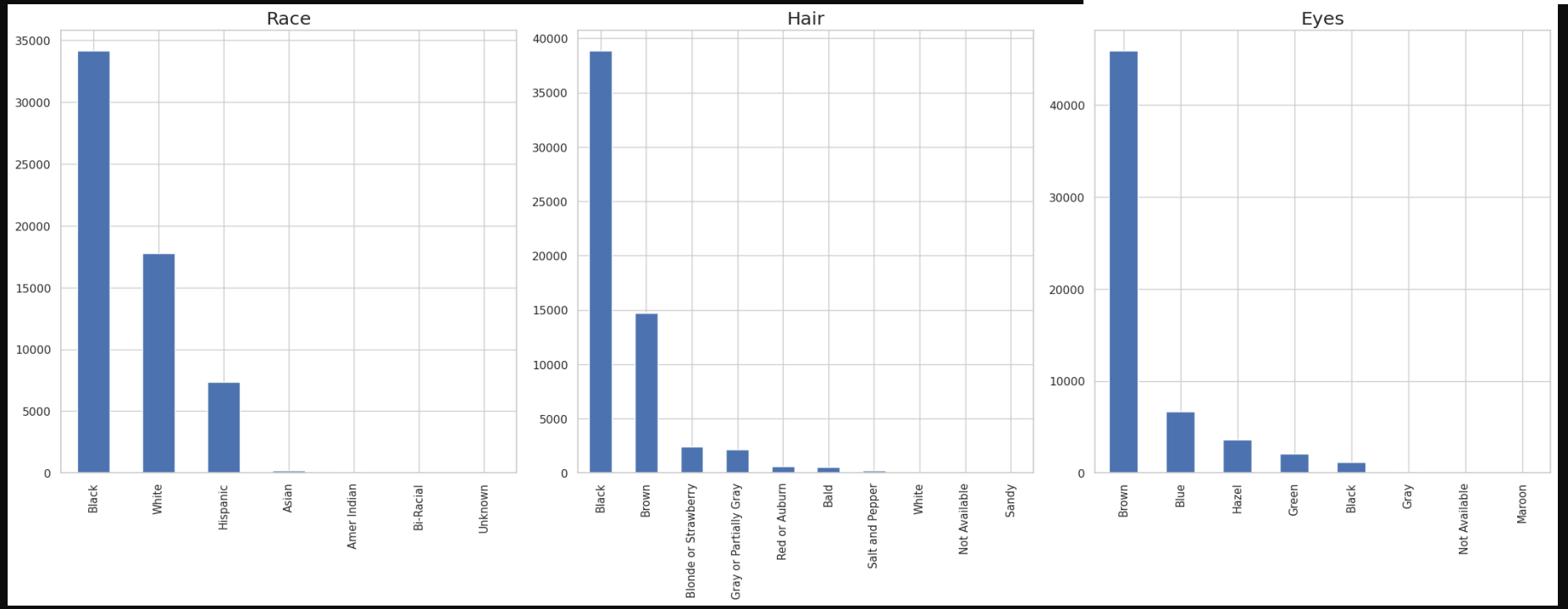
Data Pre-processing

- Removing records with invalid values, and corrupted .jpg images.
- Finding out age using datetime conversion.
- Removing outlier data.
- Converting height and weight to meters and kilograms respectively and adding the BMI feature by applying the formula.

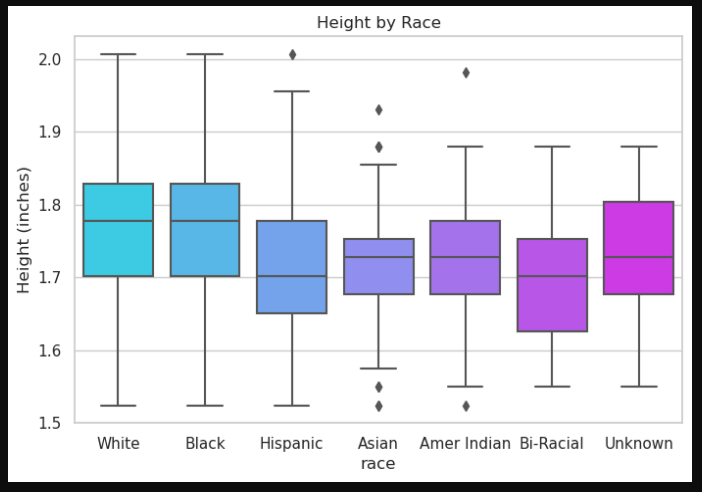


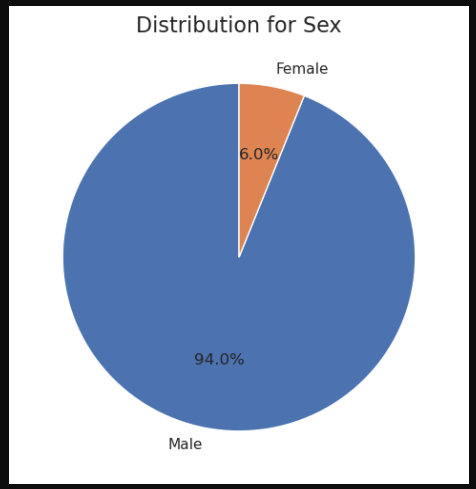
Data Visualization

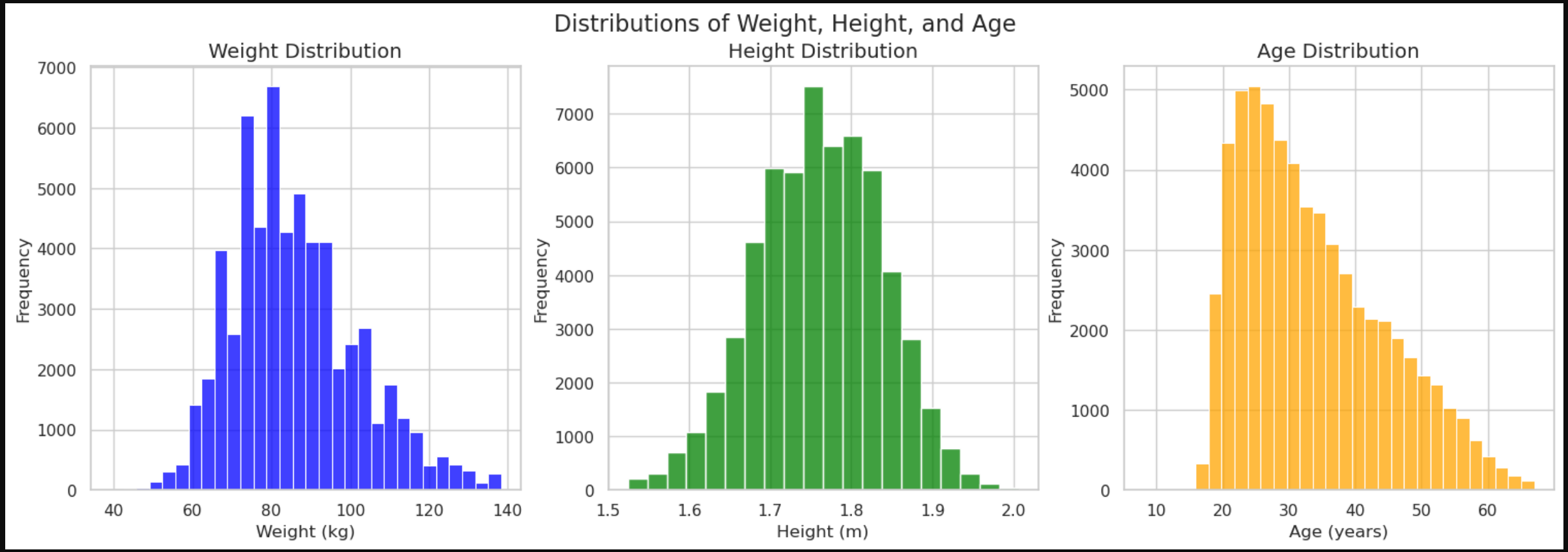




Data Visualization







Preparing Data for Model

- Preprocess Image:
- Resize to 224*224*3, convert to RGB and normalize

- Generate indices:
- Randomly Allocating indices from csv file to train and validation

```python
def preprocess_image(self, img_path):
    try:
        with Image.open(img_path) as im:
            im = im.convert("RGB")
            im = im.resize((self.IM_WIDTH, self.IM_HEIGHT))
            im = np.array(im) / 255.0
        return im
    except Exception as e:
        print(f"Error processing image {img_path}: {e}")
        return None

def generate_split_indexes(self):
    if len(self.df) == 0:
        raise ValueError("The DataFrame is empty. Cannot generate split indexes.")
    if len(self.df) < 3:
        raise ValueError("The DataFrame is too small for train/validation/test splits.")

    p = np.random.permutation(len(self.df))
    train_up_to = int(len(self.df) * TRAIN_TEST_SPLIT)
    train_idx = p[:train_up_to]
    test_idx = p[train_up_to:]
    train_up_to = int(train_up_to * TRAIN_TEST_SPLIT)
    train_idx, valid_idx = train_idx[:train_up_to], train_idx[train_up_to:]

    print(f"Train size: {len(train_idx)}, Validation size: {len(valid_idx)}, Test size: {len(test_idx)}")

    self.df['sex_id'] = self.df['sex'].map({'Male': 0, 'Female': 1})
    if self.df['sex_id'].isnull().any():
        raise ValueError("Some values in the `sex` column do not match the expected values ('Male', 'Female').")

    self.max_bmi = self.df['bmi'].max()

    return train_idx, valid_idx, test_idx
```

Preparing Data for Model

- Generate Images:
- Opening front and side images, passing them as a list of numpy arrays with bmi and gender as scalars in output list

- Create an object and use it to generate images during model training

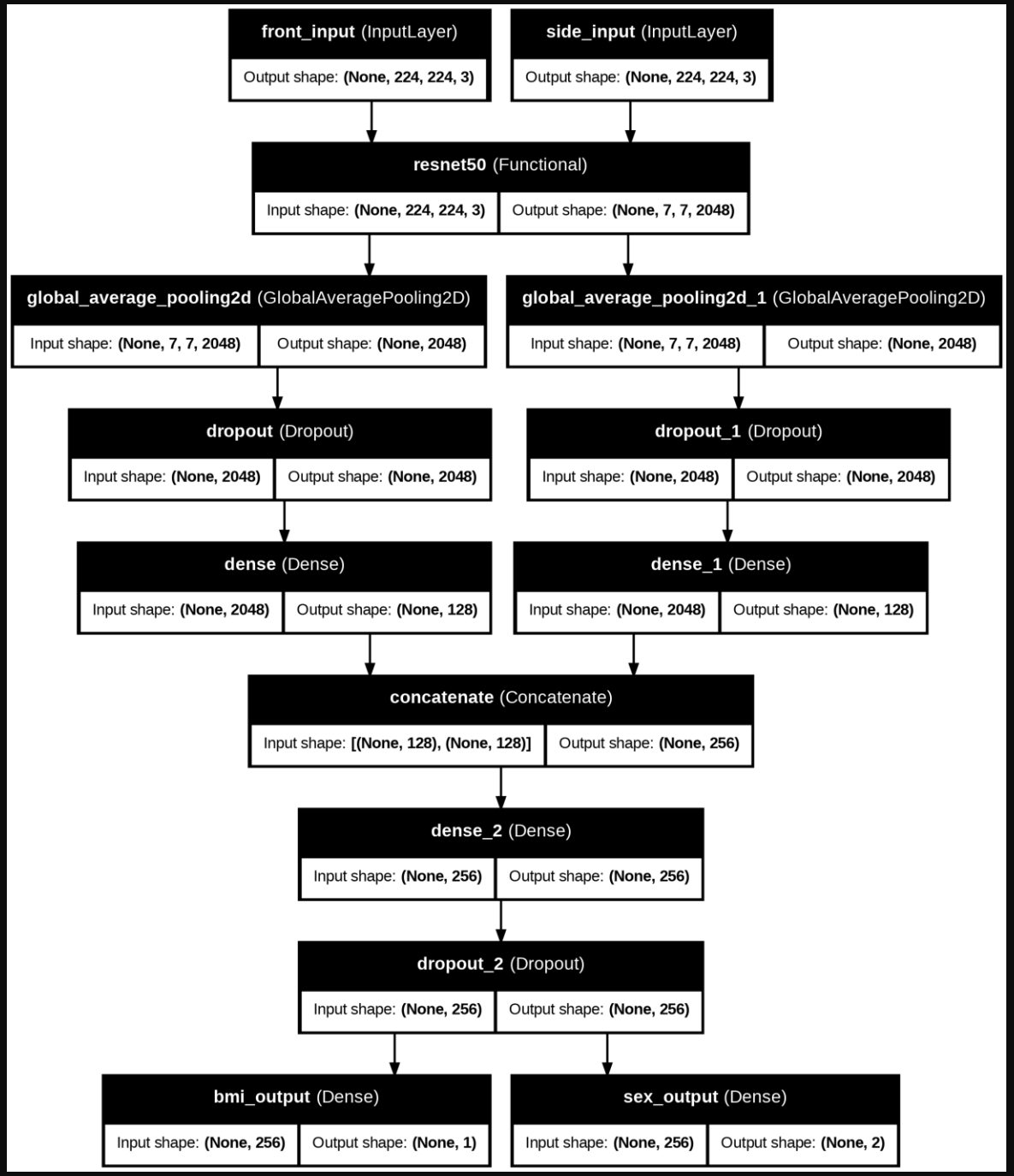
```
    def generate_images(self, image_idx, is_training, batch_size=16):
        front_images, side_images, bmis, sexes = [], [], [], []
        while True:
            for idx in image_idx:
                person = self.df.iloc[idx]

                front_file = f"/kaggle/input/illinois-doc-labeled-faces-dataset/front/front/{person['id']}.jpg"
                front_im = self.preprocess_image(front_file)

                side_file = f"/kaggle/input/illinois-doc-labeled-faces-dataset/side/side/{person['id']}.jpg"
                side_im = self.preprocess_image(side_file)

                if front_im is not None and side_im is not None:
                    front_images.append(front_im)
                    side_images.append(side_im)
                    bmis.append(person['bmi'])
                    sexes.append(to_categorical(person['sex_id'], 2))  # Assuming binary sex (0: male, 1: female)

                    if len(front_images) >= batch_size:
                        yield [np.array(front_images), np.array(side_images)], [np.array(bmis), np.array(sexes)]
                        front_images, side_images, bmis, sexes = [], [], [], []

            if not is_training:
                break

data_generator = FaceDataGenerator(df)

try:
    train_idx, valid_idx, test_idx = data_generator.generate_split_indexes()
    print("Train, validation, and test indexes generated successfully.")
except ValueError as e:
    print(f"Error: {e}")
```

```
history = model.fit(
    data_generator.generate_images(train_idx, is_training=True, batch_size=32),
    validation_data=data_generator.generate_images(valid_idx, is_training=False, batch_size=32),
    steps_per_epoch=len(train_idx) // 32,
    validation_steps=len(valid_idx) // 32,
    epochs=20
)
```

Model Architecture

- Two CNN Pipelines with pre-built ResNet50 as Base Model.
- Exclude final classification layers of ResNet50 for the purpose of custom regression and classification.
- Separate pooling, dropout and dense layers for frontal and side images.
- Fusion of both the pipelines.

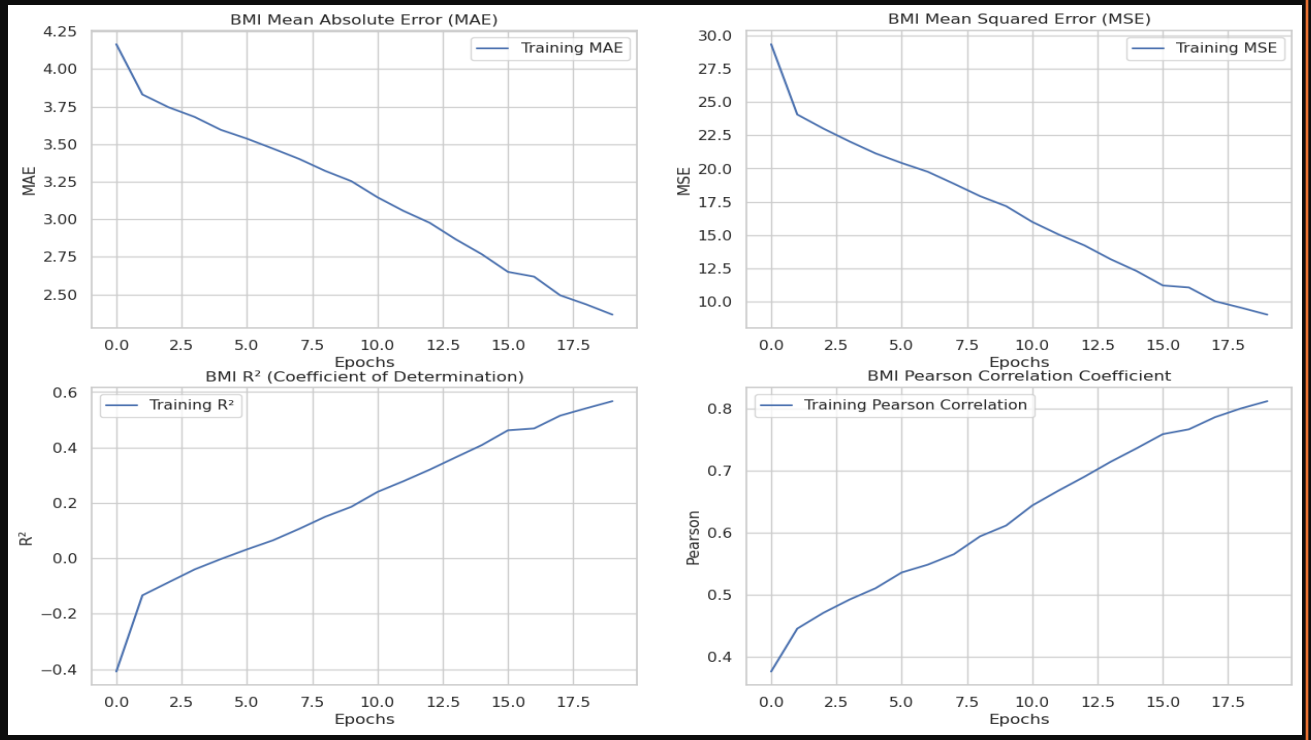
- Two dense layers:
- Regression Layer with one output for BMI
- Classification Layer using softmax regression, producing two values, denoting probability of gender.

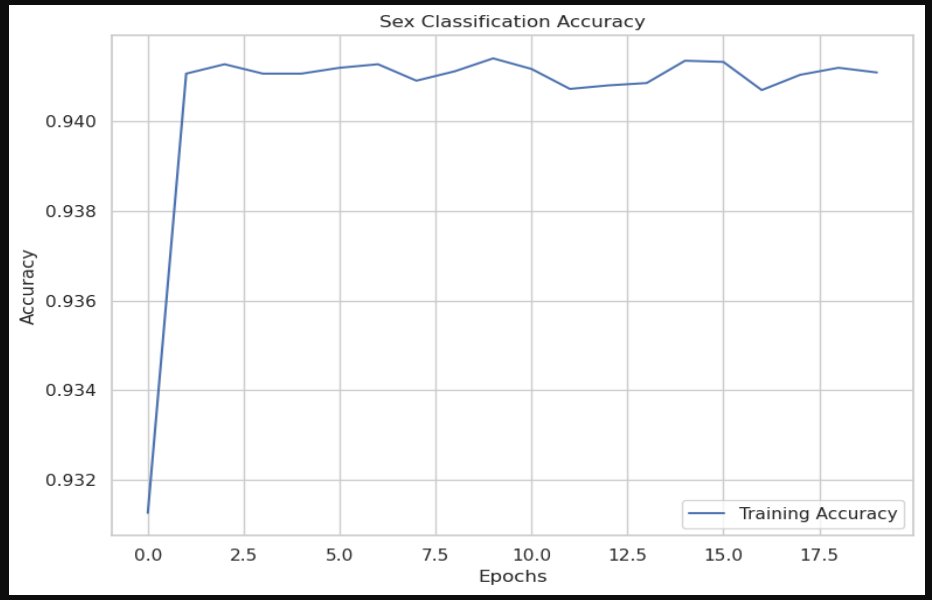


Evaluation (Training)

- Regression:
- Loss: MAE = 2.3651, MSE = 9.0137
- Metrics: R2 = 0.5672, Pearson = 0.8115

- Classification:
- Accuracy = 0.9411





Evaluation (Testing)

- Regression:
- Loss: MAE = 3.1153, MSE = 16.7522
- Metrics: R2 = 0.2788, Pearson = 0.6091

- Classification:
- Accuracy = 0.9381

Custom Image Testing

- Front and side images of teammates as input for model

Name	True BMI	True BMI Category	Predicted BMI	Predicted BMI Category	True Gender	Predicted Gender
Thribhuvan	23.67	Normal	26.21	Overweight	Male	Male
Lokesh	23.83	Normal	22.42	Normal	Male	Male
Amar	18.68	Normal	23.47	Normal	Male	Male

Conclusion

- Corrupted Images in side-view images, making it hard to pair with corresponding front-view images.
- Challenging to predict BMI using only facial images.
- Dataset having full frontal and side-view images of body could be useful.

Thank You