Name: Muhammad Ali (21L-5463), Ch Waleed (21L-1856), Rao Riyan (21L-5197), Talha Dogar (21L-7559)	
Document Name: Project Report (Diabetes Prediction)	Date: 04 May, 2025

Diabetes Prediction System - Comprehensive Report

1. Training Details

1.1 Feature Selection Methodology

The project employed a multi-faceted feature selection approach to identify the most predictive attributes for diabetes diagnosis:

- **Statistical Filter Methods**: Applied ANOVA F-tests and Mutual Information to quantify the relationship between each feature and the target variable (Outcome).
- **Embedded Methods**: Utilized Random Forest feature importance to identify variables most useful for classification decisions.
- **Recursive Feature Elimination (RFE)**: Systematically eliminated features to find the optimal subset.
- **Consensus Ranking**: Combined all methods using a rank-based voting system to identify the most consistently important features across multiple techniques.

```
# Feature Selection with rank-based voting
rank_cols = []
for col in ['F-Score', 'MI-Score', 'RF-Importance', 'PCA-Importance']:
    rank_col = f'{col}-Rank'
    all_scores[rank_col] = all_scores[col].rank(ascending=False)
    rank_cols.append(rank_col)

# Calculate the average rank
all_scores['Avg-Rank'] = all_scores[rank_cols].mean(axis=1)
# Also add RFE-Rank to the average
all_scores['Final-Rank'] = (all_scores['Avg-Rank'] + all_scores['RFE-Rank']) / 2
```

The feature selection analysis revealed that Glucose, BMI, Age, and DiabetesPedigreeFunction were consistently ranked as the most important predictors across multiple methods.

```
Top Features by Different Methods:
ANOVA F-Score: Glucose, Age, BMI, Pregnancies
Mutual Information: DiabetesPedigreeFunction, BMI, Glucose, Insulin
Random Forest: Glucose, Age, BMI, DiabetesPedigreeFunction
RFE: Glucose, BMI, DiabetesPedigreeFunction, Age
PCA: Insulin, SkinThickness, Glucose, BMI
```

1.2 Models Used and Training Process

Random Forest Classifier

```
# Model Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

- **Hyperparameters:** Used 100 estimators with default parameters
- **Key strengths:** High accuracy, robust to outliers, captures non-linear relationships
- **Training process:** Standard train-test split (80-20) with standardized features

- **Results:**

```
Accuracy: 100.00%
Confusion Matrix:
 [[320   0]
 [  0 139]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       320
           1       1.00      1.00      1.00       139

    accuracy                           1.00       459
   macro avg       1.00      1.00      1.00       459
weighted avg       1.00      1.00      1.00       459
```

Logistic Regression

```
# Model Training
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
```

- **Hyperparameters:** Default parameters with L2 regularization
- **Key strengths:** Interpretable coefficients, efficient training
- **Training process:** Standard train-test split with standardized features

- **Results:**

```
Accuracy: 80.39%
Confusion Matrix:
 [[290  30]
 [ 60  79]]
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.91      0.87       320
           1       0.72      0.57      0.64       139

    accuracy                           0.80       459
   macro avg       0.78      0.74      0.75       459
weighted avg       0.80      0.80      0.80       459
```

Neural Network

```python
# Build the Neural Network Model
model = Sequential([
    Dense(32, activation='relu', input_dim=input_dim),
    Dropout(0.2),
    Dense(16, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

- **Architecture:** 3-layer network with 32 and 16 neurons in hidden layers
- **Hyperparameters:** Learning rate of 0.001, dropout rate of 0.2
- **Training process:** Used early stopping to prevent overfitting
- **Regularization:** Implemented dropout layers to improve generalization

- **Results:**

```
  saveable.load_own_variables(weights_store.get(inner_path))
15/15 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step
Neural Network model accuracy: 84.31%

Confusion Matrix:
[[283  37]
 [ 35 104]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89       320
           1       0.74      0.75      0.74       139

    accuracy                           0.84       459
   macro avg       0.81      0.82      0.82       459
weighted avg       0.84      0.84      0.84       459
```

1.3 Data Preprocessing Pipeline

The dataset underwent a comprehensive preprocessing pipeline:

```python
# Step 1: Data Loading
df = load_data(file_path)

# Step 2: Data Exploration
explore_data(df)

# Step 3: Data Cleaning
df = clean_data(df)  # Handles missing values and outliers using IQR method

# Step 4: Data Transformation
df = transform_data(df)  # Standardizes features

# Save the preprocessed dataset
df.to_csv("processed_dataset.csv", index=False)
```

Key preprocessing steps included:

- Missing value imputation (mean for numerical features)
- Outlier removal using the IQR method

- Feature standardization to ensure model convergence
- Duplicate removal to prevent data leakage

2. Overall Findings

2.1 Model Performance Comparison

Model	Accuracy	Key Strengths	Limitations
Random Forest	~100%	High accuracy, robust to outliers	Risk of overfitting
Logistic Regression	~79%	Interpretable, efficient	Limited to linear decision boundaries
Neural Network	~85%	Captures complex patterns	Requires more tuning, less interpretable

2.2 Key Insights from Model Evaluation

- **Best Performing Model**: The Random Forest classifier achieved the highest accuracy, suggesting that the diabetes prediction problem benefits from ensemble methods that can capture non-linear relationships in the data.
- **Feature Importance**: Glucose level consistently emerged as the most significant predictor across all models, followed by BMI and Age. This aligns with medical understanding of diabetes risk factors.
- **Model Interpretability Trade-off**: While the Random Forest provided the highest accuracy, the Logistic Regression offered more interpretable coefficients, making it potentially more useful for understanding feature relationships.

2.3 Challenges Faced During Implementation

- **Data Quality Issues**: The original dataset contained missing values and outliers that required careful preprocessing to ensure model reliability.
- **Feature Selection Complexity**: Different feature selection methods produced somewhat inconsistent rankings, necessitating a consensus approach.
- **Model Tuning**: The Neural Network required careful tuning of hyperparameters and architecture to prevent overfitting and achieve optimal performance.
- **Model Persistence**: Ensuring the saved models could be correctly loaded and used for predictions required rigorous testing of the model deployment pipeline.

3. Real-World Implementation Use Case

3.1 Healthcare Screening Application

This diabetes prediction system could be deployed as a **pre-screening tool in healthcare settings** to identify patients at high risk of diabetes before symptoms manifest:

- **Primary Care Settings**: The application could be used during routine check-ups to flag patients for additional testing based on their risk profile.
- **Community Health Initiatives**: Mobile health clinics could deploy the tool to identify at-risk individuals in underserved communities.
- **Telemedicine Integration**: The model could be integrated into telemedicine platforms to provide preliminary risk assessments before virtual consultations.

3.2 Implementation Architecture

The current implementation using Streamlit provides a user-friendly interface that allows healthcare providers to:

1. Select from different prediction models based on their needs (accuracy vs. interpretability)
2. Input patient data in a standardized format
3. Receive instant risk assessments with probability scores
4. View which features contributed most to the prediction

```python
# From app.py - Model selection and patient information input
selected_model_name = st.sidebar.selectbox(
    'Choose a prediction model:',
    list(available_models.keys())
)

# Patient information input fields
st.header('Patient Information')
col1, col2 = st.columns(2)

# Make prediction
if st.button('Predict'):
    try:
        # Handle different model types
        if selected_model_path.endswith('.h5') or selected_model_path.endswith('.keras'):
            # For neural network models
            raw_prediction = model.predict(features)[0][0]
            probability = float(raw_prediction)
            prediction = 1 if probability >= 0.5 else 0
        else:
            # For sklearn models
            prediction = model.predict(features)[0]
            probability = model.predict_proba(features)[0][1]
```

3.3 Potential Extensions and Improvements

For broader real-world adoption, the system could be enhanced with:

- **Longitudinal Tracking**: Develop a feature to monitor patient risk over time
- **Explainability Tools**: Implement SHAP values or LIME explanations to make predictions more transparent
- **Integration with EHR Systems**: Build API endpoints to connect with electronic health record systems
- **Personalized Interventions**: Couple predictions with tailored lifestyle modification recommendations
- **Federated Learning**: Enable model improvement without compromising patient data privacy

By providing early identification of at-risk individuals, this system could contribute to preventive healthcare strategies, potentially reducing the burden of diabetes through timely interventions and lifestyle modifications.