Tutorial Guide - Customer Churn Prediction

Getting Started

Prerequisites

- Python 3.8 or higher
- pip package manager
- Git
- Basic understanding of machine learning concepts

Installation

1. Clone the repository

```
git clone https://github.com/saifeldeenamr10/Customer-Churn-
   Prediction-and-Analysis.git
cd Customer-Churn-Prediction-and-Analysis
```

2. Create and activate virtual environment

```
python -m venv .venv
source .venv/bin/activate  # On Unix/MacOS
.venv\Scripts\activate     # On Windows
```

3. Install dependencies

```
pip install -r requirements.txt
```

Basic Usage

1. Data Preparation

```
from project.data import DataLoader

# Load and preprocess data
data_loader = DataLoader()
X_train, X_test, y_train, y_test = data_loader.load_data()
```

2. Model Training

```
from project.models import ModelTrainer

# Initialize and train model
trainer = ModelTrainer()
model = trainer.train(X_train, y_train)
```

3. Model Evaluation

```
from project.evaluation import ModelEvaluator

# Evaluate model performance
evaluator = ModelEvaluator()
metrics = evaluator.evaluate(model, X_test, y_test)
print(metrics)
```

4. Making Predictions

```
# Make predictions
predictions = model.predict(X_test)
```

Advanced Usage

Custom Model Configuration

```
from project.models import ModelConfig

# Configure model parameters
config = ModelConfig(
    learning_rate=0.001,
    batch_size=32,
    epochs=100
)

# Train with custom configuration
model = trainer.train(X_train, y_train, config=config)
```

Feature Engineering

```
from project.features import FeatureEngineer

# Create custom features
engineer = FeatureEngineer()
X_engineered = engineer.transform(X_train)
```

Model Monitoring

```
from project.monitoring import ModelMonitor

# Set up monitoring
monitor = ModelMonitor()
monitor.start_monitoring(model)
```

Visualization Examples

Training Progress

```
import matplotlib.pyplot as plt

# Plot training history
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.show()
```

Feature Importance

```
# Plot feature importance
importances = model.feature_importances_
plt.bar(range(len(importances)), importances)
plt.title('Feature Importance')
plt.show()
```

Troubleshooting

Common Issues

1. Memory Error

- Reduce batch size
- Use data generators
- Clear memory cache

2. Training Issues

- Check data preprocessing
- Verify model architecture
- Adjust learning rate

3. **Prediction Errors**
    - Validate input format
    - Check model version
    - Verify feature scaling

Debug Tips

1. Enable verbose logging
2. Use debug mode
3. Check data types
4. Validate input shapes

Performance Optimization

Training Optimization

1. Use GPU acceleration
2. Implement batch processing
3. Optimize data pipeline
4. Use mixed precision

Inference Optimization

1. Model quantization
2. Batch inference
3. Caching predictions
4. Load balancing

Model Updates

Retraining Process

1. Collect new data
2. Update training set
3. Retrain model

4. Validate performance
5. Deploy updates

Version Control

1. Track model versions
2. Document changes
3. Maintain changelog
4. Backup models

Best Practices

Code Organization

1. Follow PEP 8
2. Use type hints
3. Write documentation
4. Add unit tests

Model Management

1. Regular evaluation
2. Performance monitoring
3. Data validation
4. Error tracking

Use Cases

Case Study 1: Churn Prediction

```
# Example churn prediction workflow
from project.churn import ChurnPredictor

predictor = ChurnPredictor()
results = predictor.predict(X_test)
```

Case Study 2: Customer Segmentation

```
# Example customer segmentation workflow
from project.segmentation import CustomerSegmenter

segmenter = CustomerSegmenter()
segments = segmenter.segment(X_test)
```

Additional Resources

Documentation

- API Reference
- Model Documentation
- Architecture Overview

External Resources

- Customer Churn Analysis
- Machine Learning Tutorial
- Model Deployment Guide