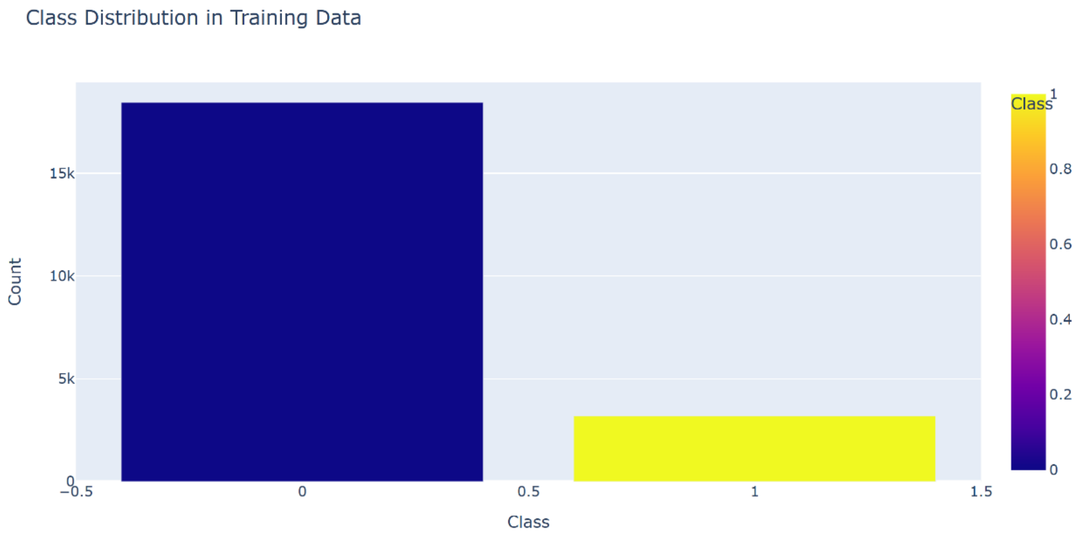
2.2 Data Cleaning & Explanatory Data Analysis

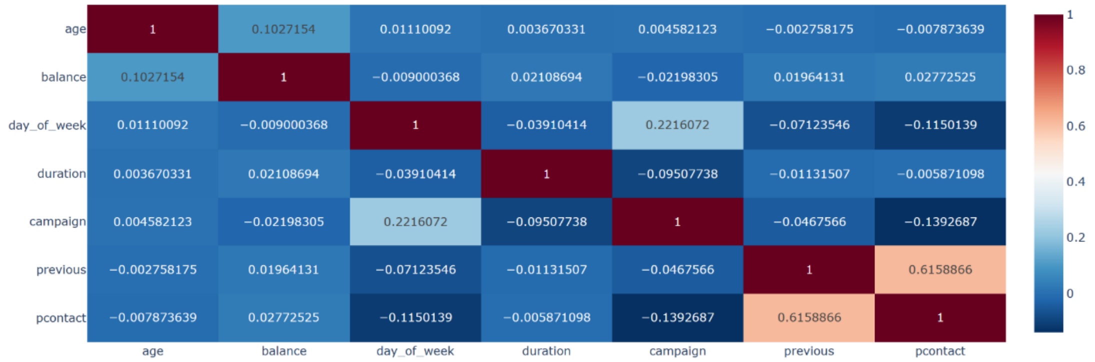
Train-test Split & Explanatory Data Analysis

```
ytrain_df = pd.DataFrame(y_train, columns=['y'])

class_counts = ytrain_df['y'].value_counts().reset_index()
fig = px.bar(class_counts,
             x='index',
             y='y',
             labels={'index': 'Class', 'y': 'Count'},
             title='Class Distribution in Training Data',
             color='index',
             color_discrete_map={'yes': 'blue', 'no': 'red'})
fig.show()
```



```
# Heatmap of all numeric features
corr_matrix = Xtrain_df.corr()
fig = px.imshow(corr_matrix, text_auto=True, aspect="auto",
color_continuous_scale='RdBu_r', title="Heatmap of Correlation
Matrix")
fig.update_xaxes(side="bottom")
fig.show()
```

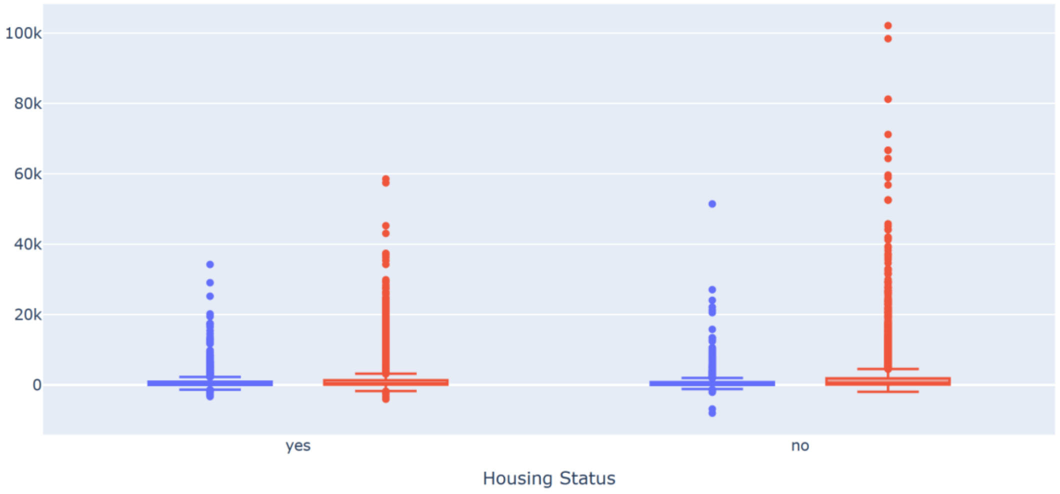
Heatmap of Correlation Matrix



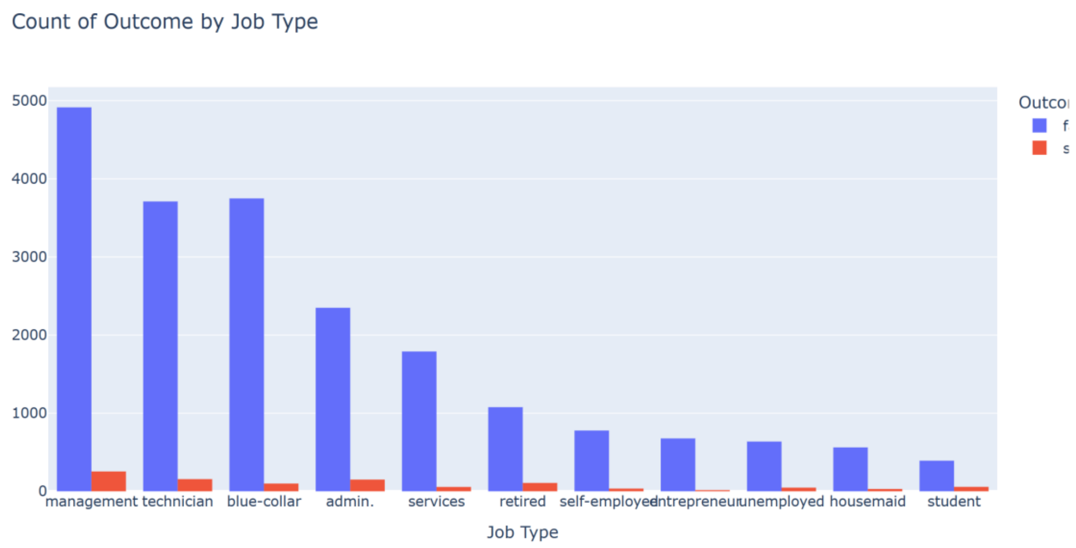
	age	balance	day_of_week	duration	campaign	previous	pcontact
age	1	0.1027154	0.01110092	0.003670331	0.004582123	−0.002758175	−0.007873639
balance	0.1027154	1	−0.009000368	0.02108694	−0.02198305	0.01964131	0.02772525
day_of_week	0.01110092	−0.009000368	1	−0.03910414	0.2216072	−0.07123546	−0.1150139
duration	0.003670331	0.02108694	−0.03910414	1	−0.09507738	−0.01131507	−0.005871098
campaign	0.004582123	−0.02198305	0.2216072	−0.09507738	1	−0.0467566	−0.1392687
previous	−0.002758175	0.01964131	−0.07123546	−0.01131507	−0.0467566	1	0.6158866
pcontact	−0.007873639	0.02772525	−0.1150139	−0.005871098	−0.1392687	0.6158866	1

```
# The relationship between balance and housing, color by loan
fig = px.box(df, y='balance', x='housing', color='loan',
            title='Balance Distribution with Housing and Loan
Status',
            category_orders={"housing": ["yes", "no"], "loan":
["yes", "no"]},
            labels={'balance':'Balance', 'housing':'Housing',
'loan':'Loan'})

fig.update_layout(yaxis_title='Balance', xaxis_title='Housing
Status', legend_title='Loan Status')
fig.show()
```

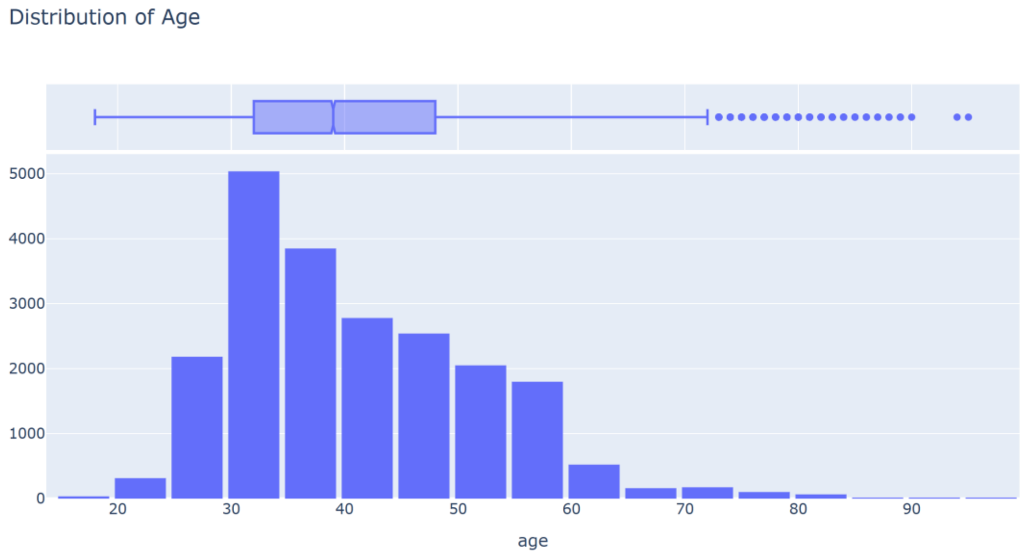
Balance Distribution with Housing and Loan Status



```
# The rank of highest job types
fig = px.histogram(Xtrain_df, x='job', color='poutcome',
barmode='group', title='Count of Outcome by Job Type',
labels={'poutcome':'Outcome', 'job':'Job Type'},
                  category_orders={"poutcome": ["failure",
"success"]})
fig.update_layout(xaxis_title="Job Type", yaxis_title="Count",
legend_title="Outcome", xaxis={'categoryorder':'total
descending'})
fig.show()
```

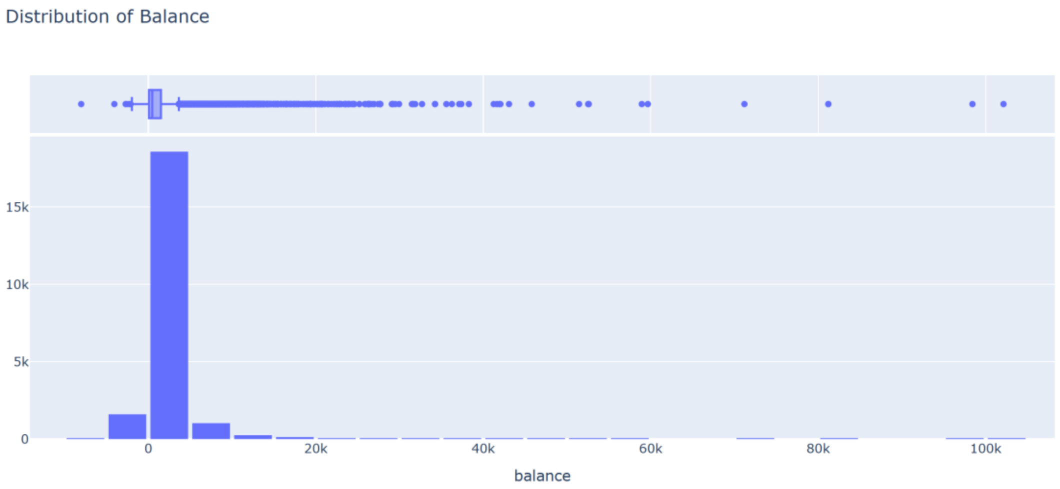


```
# Age distribution
fig = px.histogram(Xtrain_df, x='age', title='Distribution of
Age', marginal='box', nbins=30,
color_discrete_sequence=['#636EFA'])
fig.update_layout(bargap=0.1)
fig.show()
```

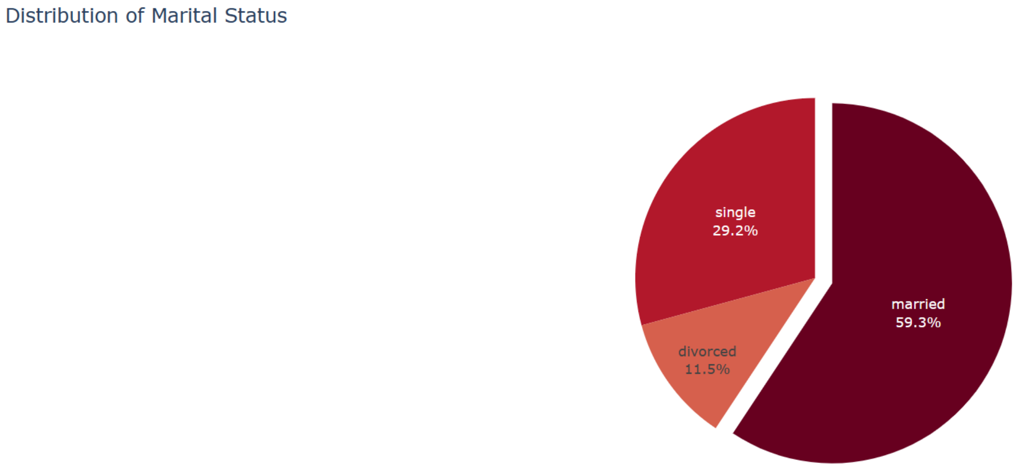


```
# Distribution of balance
```

```
fig = px.histogram(Xtrain_df, x='balance', title='Distribution of Balance', marginal='box', nbins=30, color_discrete_sequence=['#636EFA'])
fig.update_layout(bargap=0.1)
fig.show()
```

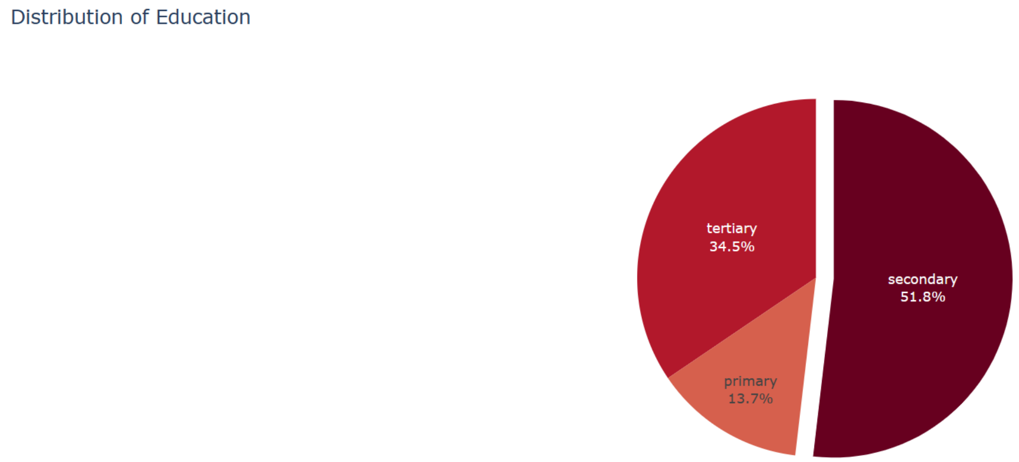


```
# Percentage of marital situation
fig = px.pie(Xtrain_df, names='marital',
             title='Distribution of Marital Status',
             color_discrete_sequence=px.colors.sequential.RdBu)
fig.update_traces(textinfo='percent+label', pull=[0.1 if i == 0 else 0 for i in range(len(df['marital'].unique()))])
fig.show()
```

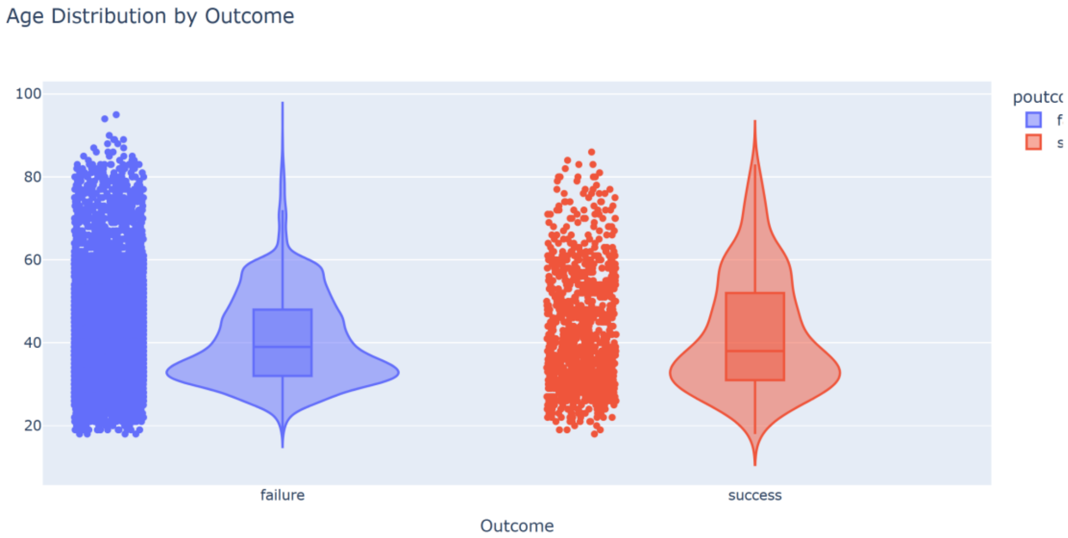


```
# Percentage of education situation
fig = px.pie(Xtrain_df, names='education',
             title='Distribution of Education',
             color_discrete_sequence=px.colors.sequential.RdBu)
```

```
fig.update_traces(textinfo='percent+label', pull=[0.1 if i == 0
else 0 for i in range(len(df['marital'].unique()))])
fig.show()
```



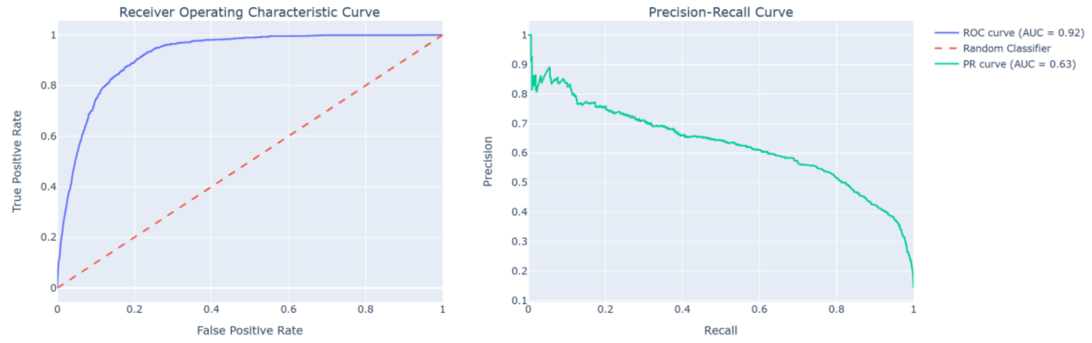
```
# (Lack of) dependence between age and outcome
fig = px.violin(Xtrain_df, y='age', x='poutcome',
color='poutcome', box=True, points="all", title='Age Distribution
by Outcome')
fig.update_layout(yaxis_title="Age", xaxis_title="Outcome")
fig.show()
```



3.7 Results of the Winner: LightGBM

```
# Get predicted probabilities for the positive class
y_scores = model_lightGBM.predict_proba(X_test_lightGBM)[:, 1]

# Compute ROC curve values
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
```

```python
roc_auc = auc(fpr, tpr)

# Compute Precision-Recall curve values
precision, recall, _ = precision_recall_curve(y_test, y_scores)
pr_auc = average_precision_score(y_test, y_scores)

# Create subplot layout
fig = make_subplots(rows=1, cols=2,
                    subplot_titles=('Receiver Operating Characteristic Curve', 'Precision-Recall Curve'))

# ROC curve
fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name='ROC curve (AUC = {:.2f})'.format(roc_auc)), row=1, col=1)
fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Random Classifier', line=dict(dash='dash')), row=1, col=1)
fig.update_xaxes(title_text="False Positive Rate", row=1, col=1)
fig.update_yaxes(title_text="True Positive Rate", row=1, col=1)

# Precision-Recall curve
fig.add_trace(go.Scatter(x=recall, y=precision, mode='lines', name='PR curve (AUC = {:.2f})'.format(pr_auc)), row=1, col=2)
fig.update_xaxes(title_text="Recall", row=1, col=2)
fig.update_yaxes(title_text="Precision", row=1, col=2)

# Update the layout and displaying the plot
fig.update_layout(showlegend=True)
fig.show()
```


Receiver Operating Characteristic Curve
True Positive Rate
False Positive Rate
Precision-Recall Curve
Precision
Recall
ROC curve (AUC = 0.92)
Random Classifier
PR curve (AUC = 0.63)